DIGITAL CIRCUIT DESIGN

Heathkit
Zenith
Educational
Systems

Written by: Andrew C. Staugaard, Jr.
Associate Professor of Electrical
Engineering & Computer Technology
Formerly of Jamestown Community College
State University of New York

Edited by: Lawrence P. Larsen
Senior Educational Media Designer
Heathkit/Zenith
Educational Systems

Model EE-1004
HEATH COMPANY
BENTON HARBOR, MICHIGAN 49022
595-3462-01

Copyright ©1985
Third Printing—1987
Heath Company
Not Affiliated with D.C. Heath
All Rights Reserved
Printed in the United States of America

CONTENTS

INTRODUCTION .... Page III
COURSE OBJECTIVES .... Page V
COURSE OUTLINE .... Page VI
PARTS LIST .... Page XV
UNIT 1 — INTRODUCTION TO DIGITAL DESIGN .... Page 1-1
UNIT 2 — COMBINATIONAL LOGIC DESIGN USING SSI DEVICES .... Page 2-1
UNIT 3 — COMBINATIONAL LOGIC DESIGN USING MSI AND LSI DEVICES .... Page 3-1
UNIT 4 — SEQUENTIAL LOGIC DESIGN .... Page 4-1
UNIT 5 — ADDING I/O TO THE DESIGN .... Page 5-1
UNIT 6 — ANALOG I/O, POWER SUPPLIES, AND DIGITAL CIRCUIT PROCESSING .... Page 6-1
UNIT 7 — DEBUGGING THE DESIGN .... Page 7-1
UNIT 8 — EXPERIMENTS .... Page 8-1
APPENDIX A — BINARY NUMBERS AND COMPUTER ARITHMETIC .... Page A-1
APPENDIX B — BASIC ELECTRONIC CONCEPTS .... Page B-1
APPENDIX C — DATA SHEETS OF ICs USED IN EXPERIMENTS .... Page C-1
INDEX .... Page I-1

Library of Congress Cataloging in Publication Data

Staugaard, Andrew C. 1947-
Digital circuit design.

"Model EE-1004 595-3462."
Includes index.
1. Digital electronics. 2. Electronic circuit design. 3. Logic design. I. Larsen, Lawrence P., 1944- . II. Heathkit/Zenith Educational Systems (Group) III. Title.
TK7868.D5S7 1985 381.5'3 85-21897
ISBN 0-87119-136-9

INTRODUCTION

Any program of study in electrical engineering technology can be divided into one of two general areas: **analog electronics** and **digital electronics**. Analog electronics involves the study of DC and AC circuits that operate on continuous analog signals. Digital electronics involves the study of logic circuits that operate on two-state digital signals. Such circuits provide the basis for decision and memory circuits that are used in digital computers. Thus, a working knowledge of digital electronics is essential for students in electrical engineering technology as well as computer science and technology.

Fortunately, you do not have to have a comprehensive background in analog electronics to understand digital electronics. The reason for this is that digital integrated circuits, or ICs, can be treated like "black boxes." That is, you are only concerned with what goes in and what comes out of the box, rather than the inner workings of the box. The inputs and outputs of digital ICs are two-state signals that represent binary 1's and 0's. So, the first prerequisite to understanding the operation of digital circuits is a knowledge of the two-state binary number system. This prerequisite material is provided in Appendix A, if you need a review or are unfamiliar with the binary number system.

You begin to need some background in electronic circuit theory when you must **interface**, or connect, digital circuits to the real **analog** world. The more complex the interface, the more analog circuit theory is required. In this course, you will be involved with simple input and output interfaces. As a result, an extensive knowledge of analog circuit theory is not required. However, you must still be knowledgeable of a few basic electronic circuit principles. This prerequisite knowledge has been provided in Appendix B, if needed.

As you can see, this course has been written to be totally independent of any other electronics courses. I have done this so that you can integrate digital electronics into any part of your electronic design education. I would suggest that you look at the material in Appendix A and B before beginning the course. If it is not familiar to you, study it and work the associated problems before you begin Unit 1.

I believe you will find this course to be an enjoyable learning experience. I have attempted to "separate the wheat from the chaff" by getting down to the important basics of digital design. You will learn **how to** design real applications type digital circuits that have been chosen to provide you with a maximum learning experience. Don't neglect the course experiments. You must "get your hands dirty" to acquire a full understanding of how digital circuits operate. The course experiments demonstrate and apply the text material. You will be instructed throughout the course when to perform these experiments.

Enjoy!

Andrew C. Staugaard, Jr.

COURSE OBJECTIVES

When you finish this course, you will be able to:

1. Describe the difference between bipolar and MOS digital integrated circuits, and the general technology used in the construction of these devices.
2. Design combinational logic circuits using SSI, MSI, and LSI ICs.
3. Design sequential logic circuits using flip-flops, buffer registers, shift registers, and counters.
4. Describe the common techniques used to input and output data to and from a digital circuit.
5. Explain the operation of digital-to-analog and analog-to-digital converters.
6. Determine the power requirements of a digital circuit and specify the appropriate power supply module.
7. Construct a working prototype of a digital circuit.
8. Explain how CAD systems are used to design digital circuits and simulate their operation.
9. Describe how to prepare a circuit for volume production from the design concept through the design process to circuit board construction and quality control.
10. Systematically troubleshoot, isolate, and repair problems within a digital circuit, that you have designed, using test instruments such as the VOM, oscilloscope, logic probe, and logic analyzer.

COURSE OUTLINE

UNIT 1 INTRODUCTION TO DIGITAL DESIGN

- Introduction
- Unit Objectives
- Basic Logic Gates
  - Digital verses Analog Signals
  - Digital Logic Gates
    - The Inverter
    - The OR Gate
    - The NOR Gate
    - The AND Gate
    - The NAND Gate
    - The XOR Gate
    - The XNOR Gate
  - Timing Diagrams
- Digital Design Considerations
  - Speed
    - Rise and Fall Time
    - Propagation Delay Time
  - Power Dissipation
  - Speed-Power Product
  - Noise Immunity
  - Fanout
  - Packing Density
  - Digital Circuit Packaging
    - TO5
    - Flat Pack
    - DIP
  - The Digital System
- Digital IC Technology
  - Bipolar Technology
    - Transistor-Transistor Logic (TTL)
    - Emitter-Coupled Logic (ECL)
    - Integrated-Injection Logic (IIL)
  - MOS Technology
    - PMOS
    - NMOS
    - CMOS
    - Precautions When Working With MOS Devices
- Summary
- Unit Examination

UNIT 2 COMBINATIONAL LOGIC DESIGN USING SSI DEVICES

Introduction
Unit Objectives
The Design Process
  Product Development
  The Black-Box Concept
  The SSI Design Algorithm
Problem Definition and Representation
  I/O Definition
  Logic Definition
  The Truth Table
  The Boolean Expression
Simplification
  Simplification Using Boolean Algebra
  Karnaugh mapping
    Two Input Variables
    Three Input Variables
    Four Input Variables
    Don't Cares
SSI Implementation
  The AND/OR Design
  The NAND/NOR/XOR Design
Summary
Unit Examination

UNIT 3 COMBINATIONAL LOGIC DESIGN USING MSI AND LSI DEVICES

- Introduction
- Unit Objectives
- Multiplexers
  - Multiplexer Types
  - Multiplexer IC Devices
    - The 74153 Dual 4-to-1 Multiplexer
    - The 74151 8-to-1 Multiplexer
    - The 74150 16-to-1 Multiplexer
    - The 74157/74158 Quad 2-to-1 Multiplexer
  - Implementing Logic Equations With Multiplexers
    - The Obvious Solution
    - The n – 1 Solution
    - The n – 2 Solution
- Decoders
  - Decoder Types
  - Decoder IC Devices
    - The 74139 Dual 1-of-4 Decoder
    - The 74138 1-of-8 Decoder
    - The 7442 1-of-10 Decoder
    - The 74154 1-of-16 Decoder
  - Implementing Logic Equations With Decoders
  - Address Decoding
  - Using Decoders as Demultiplexers
- Read Only Memories (ROMs)
  - A Simple Diode ROM
  - The ROM Symbol and Three State Buffering
  - ROM Devices
    - Mask-Programmed ROM
    - Programmable ROM (PROM)
    - Erasable/Programmable ROM (EPROM)
    - Electrically Erasable ROM (EEROM)
  - Implementing Combinational Logic Using ROMs
- Programmable Logic Arrays (PLAs)
  - The Internal Structure of a PLA
  - PLA Packages
  - PLA vs. ROM?
- Summary
- Unit Examination

UNIT 4 SEQUENTIAL LOGIC DESIGN

- Introduction
- Unit Objectives
- Flip-Flops
  - The RS Flip-Flop
  - The D Flip-Flop
    - Level vs. Edge Triggering
    - Preset and Clear
  - The JK Flip-Flop
    - Preset and Clear
    - Flip-Flop Timing Specifications
  - Some Real Devices
    - The 7474 D Flip-Flop
    - The 7475 D Flip-Flop
    - The 7473 and 7476 JK Flip-Flops
    - The 74112, 74113, and 74114 JK Flip-Flops
  - A Clock Generator IC, the 555 Timer
- Sequential Circuits: Registers and Counters
  - Buffer Registers or Latches
    - Tri-State Outputs
    - Some Real Devices
  - Shift Registers
    - Serial Input Shift Registers
    - Serial-to-Parallel Conversion
    - Parallel Input Shift Registers
    - Parallel-to-Serial Conversion
    - Some Real Counter Devices
  - Counters
    - The Ripple Counter
    - The Synchronous Counter
    - The Ring Counter
    - Some Real Devices
- Designing Sequential Circuits
  - The Sequential Circuit Design Algorithm
  - The Algorithm
    - Problem Definition
    - The State Table
    - The Input Logic
    - Output Logic
- Summary
- Unit Examination

UNIT 5 ADDING I/O TO THE DESIGN

- Introduction
- Unit Objectives
- Input Devices
  - Mechanical Switches
  - Keyboards
- Display Devices
  - Incandescent Lamps
  - Discrete LEDs
  - Segmented Displays
    - Light Emitting Diode Displays
    - Incandescent Displays
    - Gas Discharge Displays
    - Vacuum Fluorescent Displays
    - Liquid Crystal Displays
    - Decoding and Driving Segmented Displays
    - Decoding and Driving LCDs
    - Multiplexed Displays
  - Dot Matrix Displays
    - Controlling and Driving Dot Matrix Displays
- Control Devices
  - Peripheral Power Drivers
  - Transistor Arrays
  - Solid-State Relays
  - Electrical Control Devices, or Actuators
    - Electro-Mechanical Relays
    - Solenoids
  - Optocouplers
    - Optoisolator Characteristics
    - Some Typical Optoisolator Devices
- Summary
- Unit Examination

UNIT 6 ANALOG I/O, POWER SUPPLIES, AND DIGITAL CIRCUIT PROCESSING

- Introduction
- Unit Objectives
- The Analog Connection
  - Digital-to-Analog Conversion
    - Additional DAC Input Codes
    - DAC Specifications
  - Analog-to-Digital Conversion
    - ADCs
    - V/F Converters
    - Specifications and Comparisons
- The Power Supply
  - Power Supply Basics
    - Linear Supplies
    - Switching Supplies
  - Determining the Circuit Power Requirements
  - Specifying the Power Supply Module
- Digital Circuit Processing
  - Hardware Prototyping
    - Solderless Breadboards
    - Perfboard Prototypes
    - Wire Wrapping
    - Printed Circuit Boards
      - Drilling
      - Masking
    - Etching
    - Soldering
    - Inspection and Quality Control Considerations
  - Software Prototyping
    - Logic Design Using a CAD System
    - Logic Simulation, or Prototyping, Using a CAD System
    - PC Board Design Using a CAD System
  - Shielding and Grounding
    - Shielding
    - Grounding
- Summary
- Unit Examination

UNIT 7 DEBUGGING THE DESIGN

- Introduction
- Unit Objectives
- Design Bugs
  - Combinational Logic Bugs
    - Power Supply Glitches
    - Floating Inputs
    - Propagation Delay
  - Sequential Logic Bugs
    - Unclocked Inputs
    - Clock Skew
    - Excessive Clock Rates
    - Hang-up States
  - Test Instruments
    - Logic Probes
    - Logic Clips
    - Logic Monitors
    - Logic Pulsers
    - Multimeters
    - Oscilloscopes
    - Logic Analyzers
- Circuit Troubleshooting and Repair
  - Documentation
  - The Obvious
  - Testing the Power Supply
  - Testing the Clock Circuit
  - Troubleshooting Combinational Logic
  - Troubleshooting Sequential Circuits
  - Repairing the Circuit
- Summary
- Unit Examination

UNIT 8 EXPERIMENTS

Introduction
Experiment Format
Experiment 1: Fundamental Logic Gates
Experiment 2: Simple Combinational Logic Circuits
Experiment 3: Data Transmission and Parity
Experiment 4: Arithmetic Circuits
Experiment 5: Boolean Algebra
Experiment 6: Combinational Logic Design Using SSI Devices — Part 1
Experiment 7: Combinational Logic Design Using SSI Devices — Part 2
Experiment 8: MSI Devices
Experiment 9: Combinational Logic Design Using MSI Devices
Experiment 10: Combinational Logic Design Using ROM
Experiment 11: Register Circuits
Experiment 12: Counter Circuits
Experiment 13: Sequential Circuit Design
Experiment 14: Display Circuits
Experiment 15: Digital-to-Analog Conversion
Experiment 16: Analog-to-Digital Conversion

APPENDIX A BINARY NUMBERS AND COMPUTER ARITHMETIC

Introduction
Numbers and Codes
    Decimal Numbers
    Binary Numbers
Bits, Bytes, Nibbles, and Words
Binary Coded Decimal, or BCD
ASCII
Hexadecimal, or Hex, Numbers
    Decimal-to-Hex
    Hex-to-Binary
    Binary-to-Hex
Digital Computer Arithmetic
    Binary Addition
    Binary Subtraction

APPENDIX B BASIC ELECTRONIC CONCEPTS

- Introduction
- DC Circuit Basics
  - Current
  - Voltage
  - Resistance
  - Ohm's Law
  - Power
- AC Circuit Basics
  - Period and Frequency
  - Duty Cycle
  - Capacitors
- Semiconductors
  - The Diode
  - The Bipolar Transistor

APPENDIX C DATA SHEETS OF ICs USED IN EXPERIMENTS

INDEX

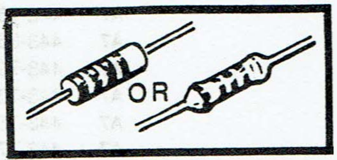
PARTS LIST

This is a list of the parts for the experiments that you will perform with this course. The key numbers correspond to the numbers on the parts illustrated in the column. Some parts are packaged in envelopes. Except for this initial parts check, keep these parts in their envelopes until they are called for in an experiment.

1/4-WATT, 5% RESISTORS

KEY No.	HEATH Part No.	QTY.	DESCRIPTION
A1	6-102-12	1	1000 Ω brown-black-red-gold
A1	6-103-12	2	10 kΩ brown-black-orange-gold
A1	6-104-12	1	100 kΩ brown-black-yellow-gold
A1	6-221-12	2	220 Ω red-red-brown-gold
A1	6-203-12	1	20 kΩ red-black-orange-gold

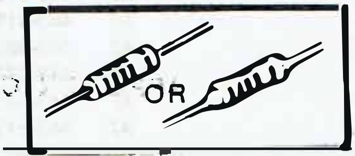
A1



1/4-WATT, 1% RESISTORS

KEY No.	HEATH Part No.	QTY.	DESCRIPTION
A2	6-1623-12	1	162 kΩ brown-blue-red-orange-brown
A2	6-3203-12	1	320 kΩ orange-red-black-orange-brown
A2	6-4002-12	1	40 kΩ yellow-black-black-red-brown
A2	6-5003-12	1	500 kΩ green-black-black-orange-brown
A2	6-8002-12	1	80 kΩ gray-black-black-red-brown

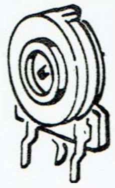
A2



POTENTIOMETER

KEY No.	HEATH Part No.	QTY.	DESCRIPTION
A3	10-312	1	10 kΩ

A3

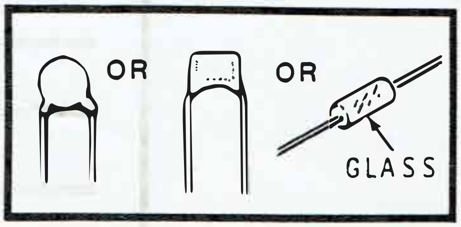


CAPACITORS

NOTE: Ceramic capacitors supplied with this course may use a new marking convention. For example, 102 = 0.001 µF, where 10 equals the capacitance in picofarads and 2 equals a powers of ten multiplier. (10×10^{-12} times $1 \times 10^{2} = 10 \times 10^{-10} = 0.001\ \mu F$.)

KEY No.	HEATH Part No.	QTY.	DESCRIPTION
A4	21-11	1	150 pF ceramic
A4	21-22	1	220 pF ceramic
A4	21-163	1	1000 pF (0.001 uF) ceramic

A4



DIODES

A5	56-16	1	1N5231B 5.1-volt zener diode
A5	56-56	8	1N4149 diode

LEDS

A6	411-875	2	5082-7731 7-segment LED

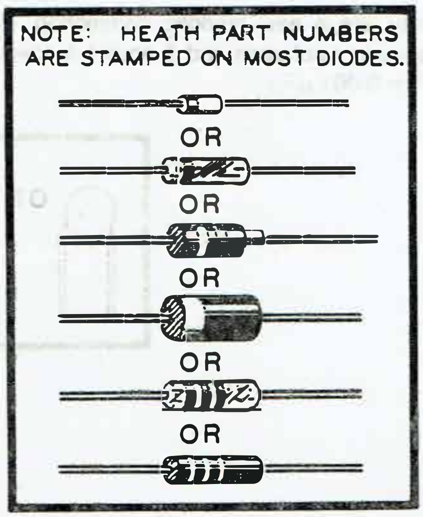
INTEGRATED CIRCUITS

A7	442-22	1	741 operational amplifier (op amp)
A7	442-753	1	9402 voltage-to-frequency (V/F) converter
A7	443-36	2	7447 7-segment decoder/driver
A7	443-728	3	74LS00 quad 2-input NAND gate
A7	443-730	2	74LS74 dual D flip-flop
A7	443-755	2	74LS04 hex inverter
A7	443-779	2	74LS02 quad 2-input NOR gate
A7	443-797	2	74LS10 triple 3-input NAND gate
A7	443-798	1	74LS20 dual 4-input NAND gate
A7	443-813	2	74LS90 decade counter
A7	443-815	1	74LS193 4-bit up/down counter
A7	443-829	2	74LS76 dual JK flip-flop
A7	443-877	1	74LS138 decoder/demultiplexer
A7	443-891	2	74LS86 quad 2-input XOR gate
A7	443-949	1	ADC0804 analog-to-digital (A/D) converter
A7	443-955	1	74LS153 multiplexer

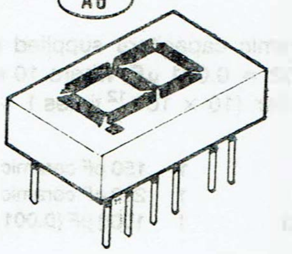
MISCELLANEOUS

	344-59	20′	#22 white solid wire
	266-962	1	5-compartment container
	597-260	1	Parts Order form

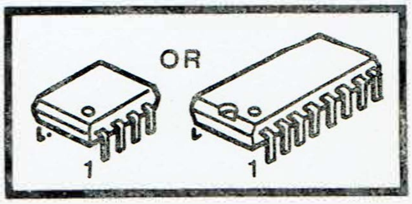
A5



A6



A7



UNIT 1

INTRODUCTION TO DIGITAL DESIGN

INTRODUCTION

As the name implies, this first unit will be an introductory unit that provides a degree of continuity between any prerequisite material and the material in this course. At a minimum, you should be familiar with binary numbers and computer arithmetic, and basic electronic terms. If you are not, refer to Appendix A and Appendix B before you proceed.

This unit begins with a discussion of the fundamental building blocks of all digital electronics — digital logic gates. You will want to pay close attention to this material, since the concepts presented here are required for the remainder of the course. Even if you are already familiar with this material, a good review will not hurt.

The second section in this unit is extremely important, since it discusses digital design considerations. The design considerations presented here relate to various digital IC technology options available to you, the designer.

Finally, the last section in this unit provides an overview of the various digital IC technology devices readily available on the commercial market. Each series of devices is discussed as related to the design considerations presented in the previous section. You will want to pay particular attention to the characteristics of each series of digital IC devices. This will allow you to make prudent decisions when you must specify the type of ICs to be used in your design.

9. Define the following logic gate specifications:

   A. Rise Time
   B. Fall Time
   C. Propagation Delay
   D. Speed-Power Product
   E. Noise Margin

10. List the four major design levels of a digital system.

11. Explain the difference between SSI, MSI, LSI, VLSI, and SLSI.

12. Explain what is meant by the terms IC "family" and IC "series."

13. State the general characteristics of the TTL, ECL, I^2L, PMOS, NMOS, and CMOS IC families.

14. Describe the differences between 74, 74L, 74H, and 74LS TTL devices.

15. Explain how to make connections between TTL and CMOS devices.

16. State the precautions that must be taken when working with MOS devices.

BASIC LOGIC GATES

Before you can attempt to design working digital circuits, you must be familiar with the fundamental building blocks of digital electronics. These devices are called logic gates and provide the foundation upon which all digital circuits, including computers, are built.

In this section, you will be introduced to the seven basic gates of digital electronics. You will find that all digital circuits, no matter how complex, can be reduced to these seven basic gates. However, before we begin our description, it might be helpful to review the difference between digital and analog signals.

Digital vs. Analog Signals

Most of the world that you are familiar with is an analog world. The sun gradually rises in the morning then gradually sets in the evening. It doesn't turn on in an instant like a light bulb in the morning then turn off in the evening. Likewise, the temperature gradually increases during the day and decreases toward evening. Natural, real world processes such as these are analog processes. That is, they produce gradual, rather than abrupt, changes. If you were to use a voltage to represent the sunlight or temperature during the day you would find that the voltage would gradually increase to a maximum level then gradually decrease to some minimum level. Such a voltage is called an analog signal.

ANALOG SIGNAL: A continuous voltage or current signal that can assume an infinite number of values between some minimum and maximum level.

You might suggest that smaller voltage levels be used. However, this creates a noise problem. If numbers were represented with millivolt levels, electrical noise from motors, lights, appliances, etc. would interfere with the analog signals and cause false numbers to appear in the circuit.

Other problems associated with analog computers are speed and size. An analog switching circuit takes a relatively long time to go from a very small to a very large value. Thus, analog computers are relatively slow as compared to digital computers. In addition, analog circuits require relatively large and accurate power supplies to accurately represent a wide range of numerical values. This creates both a size and accuracy problem.

The solution to the inherent disadvantages of using analog signals to represent numbers is found in the two state digital system.

DIGITAL SIGNAL: A voltage or current that can only assume one of two levels (see Figure 1-2).

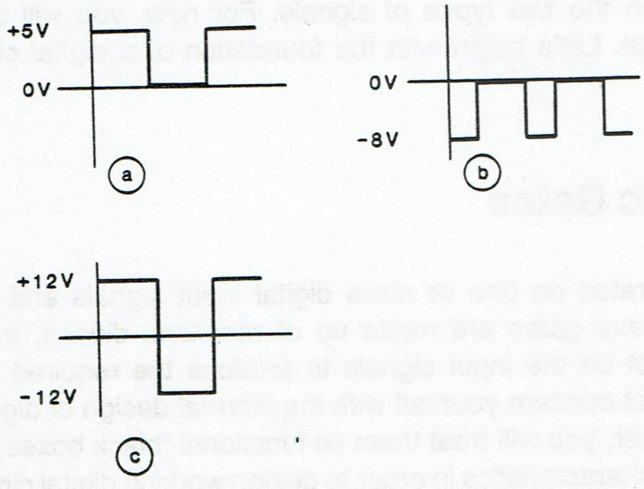


Figure 1-2
Types of digital signals.

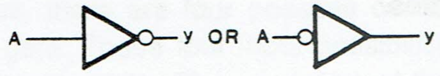
Any digital device or circuit can be described in one of the three following ways:

1. A symbol or set of symbols that are used to form a schematic diagram.
2. A truth table.
3. A Boolean expression.

The symbol provides a graphical representation of the device for circuit diagramming purposes. The truth table describes the device operation by providing a list of all the binary input possibilities and the associated output for each input value. Lastly, the Boolean expression provides a mathematical description of the device or circuit. The following discussion will employ all three representation techniques to describe the respective logic gates.

THE INVERTER

The inverter is the simplest of all logic gates. It has one input and one output. The output is always the opposite of the input. Thus, a logic 1 output is produced by a logic 0 input, and vice-versa. The associated symbol, truth table, and Boolean expression are provided in Figure 1-3.



ⓐ SYMBOLS(s)

A	y
0	1
1	0

ⓑ TRUTH TABLE

$y = \overline{A}$

ⓒ BOOLEAN EXPRESSION

Figure 1-3
An inverter symbol **a**, truth table **b**, and Boolean expression **c**.

Notice that the input is NOTed twice by the two inverters. As a result, the Boolean expression must be: $y = \overline{\overline{A}}$. From observing the truth table, you can conclude that y = A is equivalent to this expression. In symbols, $\overline{\overline{A}} = A$.

It might seem that the circuit in Figure 1-4 doesn't do much, since its logic output equals its input. However, such a circuit is sometimes used to boost signal levels in a signal path without affecting the logic of the signal. When this is done, the circuit or device is called a **buffer**.

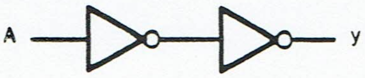
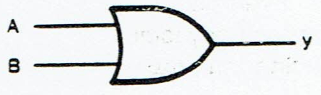


Figure 1-4
Two inverters connected in series (Example 3-1).

THE OR GATE

The symbol, truth table, and Boolean expression for the OR gate is shown in Figure 1-5. The first thing that you might note is that this particular OR gate has two input lines. Since each input line can supply a binary bit, the gate shown in Figure 1-5 is called a 2-bit, or 2-input, OR gate. There are two input lines to the gate. As a result, there are four possible combinations of 0 and 1 that can be applied to the gate. These four input possibilities are shown on the left side of the associated truth table. The operation of the gate can be explained by looking at the corresponding gate outputs on the right side of the table. Notice that the gate generates a logic 1, or high, output whenever one "or" more of its inputs are high.



ⓐ SYMBOL

A	B	y
0	0	0
0	1	1
1	0	1
1	1	1

ⓑ TRUTH TABLE

$y = A + B$, $y = A \vee B$

ⓒ BOOLEAN EXPRESSIONS

Figure 1-5
The OR gate.

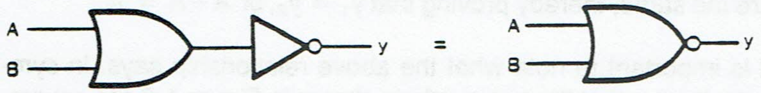
The output of the 3-bit OR gate is determined by simply remembering that the output of an OR gate is high whenever one or more of its inputs are high.

Extending the 2-bit OR gate Boolean expression to accommodate three inputs simply requires the addition of another OR (+) symbol along with the third input. Thus, the 3-bit OR gate expression becomes: $y = A+B+C$.

You might expect that an OR gate can have any number of inputs, and you are right. You now have the knowledge to construct a symbol, truth table, and Boolean expression for an OR gate with any number of input lines.

THE NOR GATE

NOR stands for "not OR." Thus, to get a NOR gate, you simply add an inverter to the output of an OR gate as shown in Figure 1-7a. Since the "not OR" operation is so common, it has its own symbol. Notice that the NOR symbol simply adds a bubble to the output line of an OR gate symbol. Note that you cannot "not" the inputs to an OR gate and produce a NOR gate. You actually produce a different form of gate. This will be described later. The corresponding NOR gate truth table, shown in Figure 1-7b, is simply the OR gate truth table with the output inverted. Likewise, the Boolean expression for a NOR gate places a bar over the OR gate expression as shown in Figure 1-7c.



ⓐ SYMBOL

A	B	y
0	0	1
0	1	0
1	0	0
1	1	0

ⓑ TRUTH TABLE

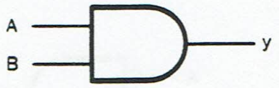
$y = \overline{A+B}$, $y = \overline{A \vee B}$

ⓒ BOOLEAN EXPRESSION(S)

Figure 1-7
The NOR gate.

THE AND GATE

The symbol, truth table, and Boolean expression for a 2-bit AND gate are shown in Figure 1-9. From the truth table, you can see that the output of an AND gate is high when **all** of its inputs are high. Thus, the output, y, is high when both A "and" B are high.



ⓐ SYMBOL

A	B	y
0	0	0
0	1	0
1	0	0
1	1	1

ⓑ TRUTH TABLE

$y = A \bullet B = AB = A \wedge B$

ⓒ BOOLEAN EXPRESSIONS

Figure 1-9
The AND gate.

The Boolean symbol used to denote the ANDing operation is the dot, or "•", as shown in Figure 1-9c. This symbol comes from the idea that, to get the correct output you can think of the ANDing operation as a multiplication operation. For example, if $A = 1$ and $B = 1$, then $A \bullet B = 1 \bullet 1 = 1$. From the truth table, you can see that this is the output that results from an input value of 1 AND 1. However, ANDing should not be confused with multiplication, since they are two different operations. ANDing is a logic operation, while multiplication is an arithmetic operation. To avoid confusion, some computer texts use the symbol "$\wedge$" to denote an ANDing operation. In fact, some texts drop all symbols and simply write $y = A \bullet B$ as $y = AB$. From now on, I will use this method.

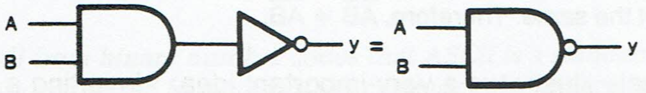
From the above substitutions, the truth table for $y_1 = \overline{A}\,\overline{B}$ is:

A	B	y_1
0	0	1
0	1	0
1	0	0
1	1	0

You should recognize that this is the truth table for a NOR gate. Thus, $y_1 = y_2$, or $\overline{A}\,\overline{B} = \overline{A+B}$. In words, a bubbled AND gate performs the same logic operation as a NOR gate. This is a very important relationship in Boolean algebra known as **DeMorgan's first theorem.** You will use it later to analyze digital circuits.

THE NAND GATE

Just like NOR stands for "not OR," NAND means "not AND." So, to get a NAND gate, you simply add an inverter to the output of an AND gate. As a result, the NAND gate truth table shown in Figure 1-11B is an AND gate truth table with the outputs inverted. Likewise, the Boolean expression for a NAND gate places a bar over the AND gate expression to invert the expression as shown in Figure 1-11C.



ⓐ SYMBOL

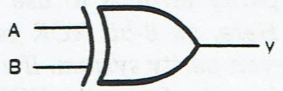
A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

ⓑ TRUTH TABLE

$y = \overline{AB}$, $y = \overline{A \wedge B}$

ⓒ BOOLEAN EXPRESSIONS

Figure 1-11
The NAND gate.



ⓐ SYMBOL

A	B	y
0	0	0
0	1	1
1	0	1
1	1	0

ⓑ TRUTH TABLE

$y = A \oplus B, \quad y = A \veebar B$

ⓒ BOOLEAN EXPRESSIONS

Figure 1-12
The XOR gate.

An odd number of 1's in a binary word is called **odd parity**. Conversely, an even number of 1's in a binary word is called **even parity**. As you might imagine, an XOR gate can be used to "recognize" odd parity, but not even parity.

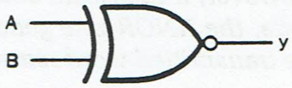
Example 1-6

Recall from binary number codes that ASCII is a standard 7-bit binary code used to represent the common typewriter keyboard characters. Although ASCII is a 7-bit code, eight bits are used when transmitting ASCII characters. An extra, or eighth, bit is added to the 7-bit ASCII code so that transmission errors can be detected. This eighth bit is called a ***parity bit****.*

A logic 1 or 0 is placed in the most significant, or parity, bit position such that the transmitted 8-bit code is always one parity, even or odd. If a bit is changed during transmission, it changes the parity of the code. Such bit errors are usually caused by electrical noise from motors, generators, and atmospheric disturbances, just to mention a few sources. When a receiving device reads the transmitted code, it must first check to see if the parity is correct. In an even parity system, the binary word of code will be even parity if no errors have occurred. Consequently, an odd parity word in an even parity system represents an error condition.

THE XNOR GATE

If the output of an XOR gate is inverted, what does it become? You guessed it, it becomes an XNOR gate. The corresponding symbol, truth table, and Boolean expression are provided in Figure 1-14. Observe from the truth table that the XNOR gate recognizes even parity. Consequently, an even number of 1's (or all 0's) applied to the gate produces a high output.



ⓐ SYMBOL

A	B	y
0	0	1
0	1	0
1	0	0
1	1	1

ⓑ TRUTH TABLE

$$y = \overline{A \oplus B}, \quad y = \overline{A \veebar B}$$

ⓒ BOOLEAN EXPRESSIONS

Figure 1-14
The XNOR gate.

In other words, a timing diagram is a picture of the digital circuit's operation. It is what you would expect to see if a multi-channeled oscilloscope were connected to the inputs and outputs of a digital circuit.

A timing diagram for a common AND gate application is shown in Figure 1-16. Notice how the timing diagram allows you to immediately visualize the operation of the gate. This particular diagram shows that the gate output is high whenever both of its inputs are high. You might be wondering why a timing diagram was used to show this, when a truth table could be used to show the same thing. The truth table only shows an instant in time while a timing diagram shows a period of time. Thus, the timing diagram provides a more general illustration of the gates operation. It shows you that the CLK input to the gate is seen at the gate's output only when the CONTROL input is high. In other words, the CLK input is passed by the gate whenever CONTROL is high. You could say that the circuit operation is being controlled by the CONTROL input. For obvious reasons, such an input is often called a **control line**.

Given the input signals to a digital circuit and knowing the operation of the circuit, you can easily sketch a timing diagram for the circuit. The trick is to look at the inputs at each point in time and, knowing the operation of the circuit, plot the corresponding output.

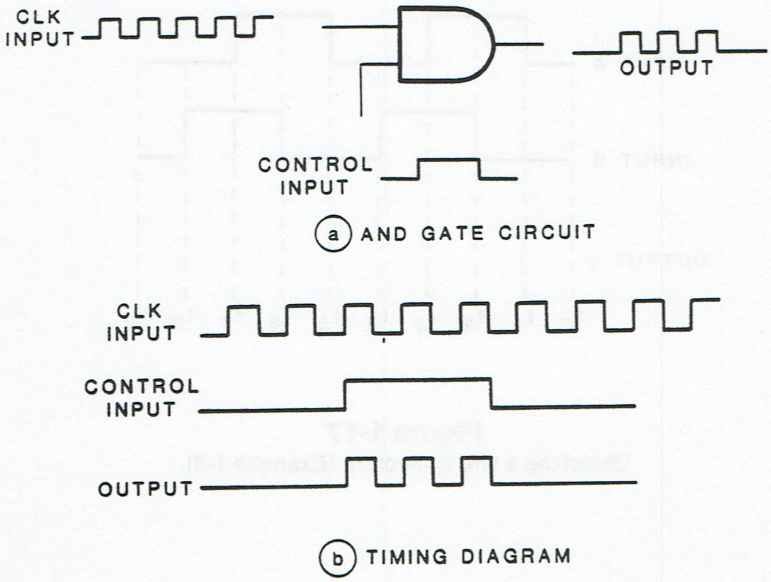


Figure 1-16

A timing diagram provides a picture of the digital circuit's operation.

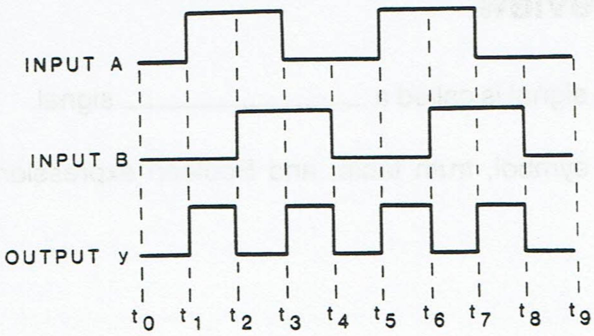


Figure 1-18
The completed timing diagram for Example 1-8.

The timing diagram obtained in Example 1-8 shows another important application of an XOR gate. Looking at the completed timing diagram in Figure 1-18 you can see that the output waveform of the XOR gate is high in proportion to the phase relationship between the input waveforms. If the two input waveforms were in-phase, or in-step, with each other, the XOR gate output would be a constant logic 0. On the other hand, if the two input waveforms were completely out of phase with each other, the gate output would be a constant high level. Any in-between phase condition produces an output that is high in proportion to the phase difference between the two input waveforms. In this application, the XOR gate is acting as a **phase comparator**.

The above application shows the importance of timing diagrams. Without the timing diagram, it is very hard to see how the gate acts as a phase comparator. This is why timing diagrams are an extremely important tool for digital circuit design and analysis. We will leave timing diagrams for now. However, you will see them again when you begin to design digital circuits.

6. A bubbled-OR gate is equivalent to a common __________ gate.

7. $\overline{A + B} = \overline{A} + \overline{B}$. ______________
True/False

8. The Boolean expression for the circuit in Figure 1-19 is:

____

9. The truth table for the expression in question #8 shows that it is equivalent to a common ____________ gate?

10. A ________________ gate must be used as a parity bit generator in an even parity system.

11. Suppose $A_3A_2A_1A_0$ = 0101 in Figure 1-20. What does $y_3y_2y_1y_0$ equal when:

A. COMPLEMENT = 0? ____________
B. COMPLEMENT = 1? ____________

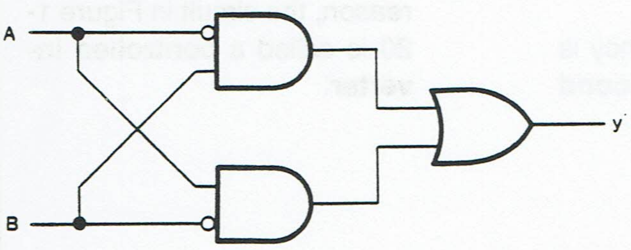


Figure 1-19
Review Question 8.

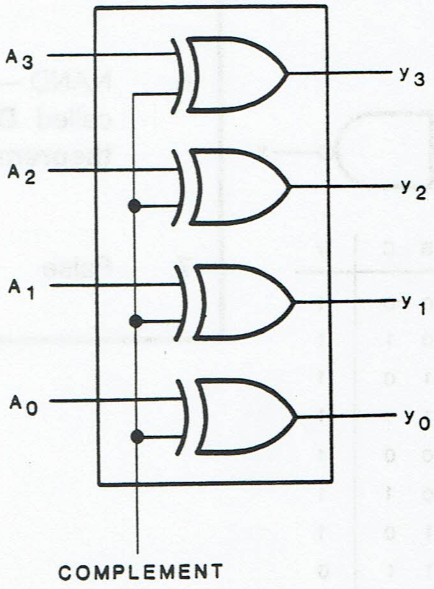


Figure 1-20
Review Question 11.

DIGITAL DESIGN CONSIDERATIONS

The logic gates discussed in the last section are manufactured into electronic devices called **integrated circuits, or ICs**. A digital integrated circuit is a semiconductor device which combines the operation of several electronic components onto a single "chip" to perform a digital operation. Because of integrated circuits, digital equipment can be designed almost entirely from a functional logic level, without much concern for the internal components and circuits of the IC. As a result, digital ICs can be treated as black-box type building blocks during the design process. Thus, it is only important that you know the functional characteristics of the IC.

Aside from the logic operation of a digital IC, several "real world" characteristics of digital devices must be considered in order to design working digital circuits. These characteristics include:

- Speed
- Power Dissipation
- Fan Out
- Noise Immunity
- Density
- Packaging
- Cost

As you will soon discover, different types of digital ICs have different design, or operating, characterisitics. Thus, your job as a circuit designer is to pick the type of digital IC that best suits the application requirements.

In this section, you will investigate each of the above design considerations. Then, in the next section, you will be acquainted with several different types of digital ICs and see how each type of IC has it own advantages and disadvantage as related to the above characteristics. This will allow you to make prudent decisions on what type of ICs to use for a given application.

in Figure 1-24. Note that the output transition occurs at a specific time after the input transition. This is the propagation delay of the device. Notice also that propagation delay is measured between the 50% levels of the input and output signals and there are two propagation delays. One, labeled $\mathbf{t_{PLH}}$, is when the input causes the output to change from low-to-high. The other, $\mathbf{t_{PHL}}$, is when the input causes the output to change from high-to-low. Propagation delay for most digital logic gates is in the nanosecond (10^{-9}) range.

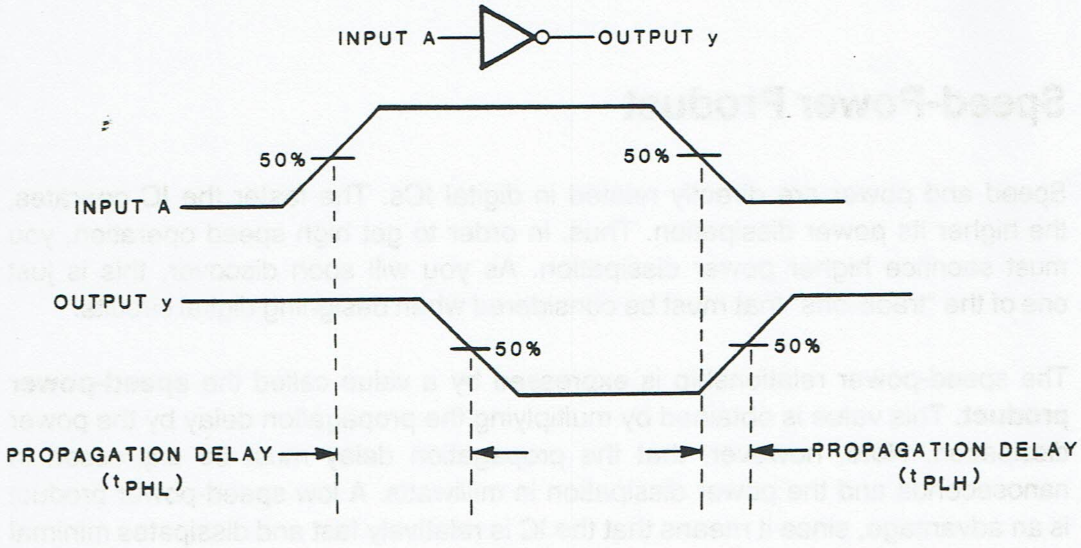


Figure 1-24
Propagation delay, t_p.

Power Dissipation

All electronic devices consume a certain amount of power when they operate. Most of this power is dissipated in the form of heat. Since excess heat is a major cause of failure in digital circuits, you must pay close attention to power dissipation during the design process. Of course, the total power consumed and amount of heat dissipated is a direct function of the number of devices in the circuit. Thus, the total power dissipation of the digital circuit will determine the size and cost of the power supply, as well as any special cooling requirements for the circuit. Of particular importance is the power dissipation in a battery operated circuit. Low power dissipation reduces the size and cost of the battery, and ensures long battery life.

Noise Immunity

The immunity that a digital circuit has to electrical noise is especially important to industrial applications due to electrical interference from heavy machinery. Even in non-industrial applications, a substantial amount of electrical noise is generated by the switching action of the digital circuit itself. Electrical noise can cause faise triggering of the logic circuitry, resulting in "glitches" in the logic signal path. Figure 1-25 shows how noise can effect a digital signal. If this were an input to a logic gate, the gate might interpret the noise as a series of 1's and 0's and trigger on the noise rather than the actual digital signal.

The noise immunity of a digital device is a measure of its ability to **reject** electrical noise. All digital logic circuits have built-in noise immunity. The noise immunity of most logic circuits is between 10 and 50 percent of the supply voltage for the device. This means that a noise spike occurring on a logic 0 or logic 1 level will be rejected if its amplitude is below a level that is 10 percent to 50 percent of the supply voltage. Consequently, if a 5 volt digital IC has a noise immunity factor of 10 % it will reject any electrical noise that is less than .5 volt in amplitude. In other words, noise spikes that are less than .5 volt will not affect the operation of the circuit. Noise immunity will be discussed in more detail in the next section were it can be related to specific types of digital ICs.

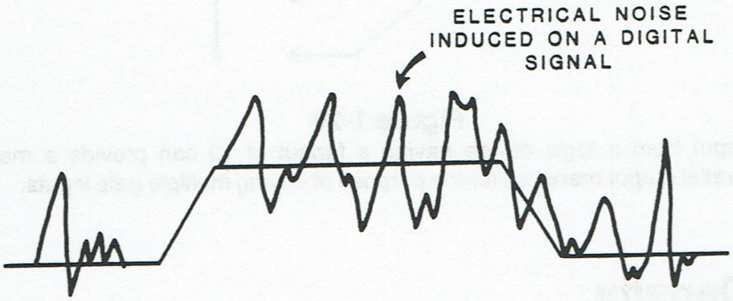


Figure 1-25
The effect of electrical noise on a digital signal.

When the transistor was invented at Bell Telephone Laboratories in 1948, scientists and engineers were able to duplicate the operation of a vacuum tube in a small piece of silicon called a chip. A single chip containing one integrated transistor is called a **discrete** transistor device. Discrete transistors are still with us today, especially for higher power applications.

Several years later, in 1961 at Fairchild Camera and Instrument Co. and concurrently at Texas Instruments, scientists and engineers developed a process called **photolithography** that allowed the packing of more than one transistor within a single silicon chip. This was the birth of the integrated circuit, or IC. The integrated circuit was not a new invention like the transistor, rather it represented a processing breakthrough that would eventually lead to computers on a chip.

The term **small-scale integration, or SSI,** is used to describe the density level of these first ICs. Digital technology was born as a result of SSI, since several transistors could be integrated into a single chip to form digital logic gates. Small-scale integration allows up to 12 gates to be integrated within a single chip.

As processing technology became better in the mid-1960's, more and more transistors could be packed within a single chip. As a result, the term **medium-scale integration, or MSI**, was created to describe digital ICs containing from 12 to 100 gates per chip. MSI circuits have multiple gates which are internally interconnected to form a complete functional circuit, such as a counter, decoder, or multiplexer. These devices will be discussed in subsequent units. Such circuitry eliminates the need of having to interconnect individual gates from SSI devices to form the same function. Thus, the **chip count** of a circuit can be reduced, thereby reducing cost, assembly time, and improving the reliability of the circuit.

In the early 1970's, improved IC processing technology led to the development of **large scale integration, or LSI**, whereby more than 100 gates are packed within a single chip. The microprocessor is a direct result of LSI circuit technology.

Today, a processing technique known as **electron beam lithography** allows thousands of gates to be integrated within a single silicon chip. This technology has led to **very large scale integration, or VLSI**, and the development of complete computers on a single chip. Someday, VLSI will give way to **super large scale integration, or SLSI**, involving tens, even hundreds of thousands of gates integrated into a single silicon chip, leading into an age of super fast and intelligent computers.

FLAT PACK

The flat pack was another type of housing used in the early development stages of integrated circuits. It is the smallest of all available integrated circuit packages and is designed for high density packaging. The packages are flat and are designed to be soldered or spot welded to a circuit board. Circuits can be placed close together and, therefore, a considerable amount of circuitry can be packaged in an extremely small area. Because of their ability to be packed so densely, these integrated circuits are generally made of a ceramic material that can withstand high temperatures. Closely packaged circuits cause heating and cooling problems. Therefore, the circuitry must be able to withstand such an environment. Flat pack circuits are used primarily in critical-size applications such as avionics, high reliability military systems, and special industrial equipment.

DUAL IN-LINE PACKAGE

The most widely used form of integrated circuit packaging is the dual in-line package, or DIP. It is slightly larger than the other types available, but it offers many advantages. Such circuits are easy to mount and use. They are designed to be adaptable to machine insertion on printed circuit boards. In addition, they are available in various sizes, all the way from an 8-pin package (mini DIP) to a 64-pin package. Most SSI circuits are housed in 8-, 14-, or 16-pin dual in-line packages. MSI circuits are found in 14-, 16-, 18-, 20-, 22-, and 24-pin dual in-line packages. LSI circuits, because of their greater size and complexity, require a greater number of input and output leads and, therefore, are usually housed in 24-, 28-, 40-, and even 64-pin packages.

Several different types of dual in-line package materials are used. The most commonly used and least expensive is a plastic package. In this type of package the integrated circuit chip is spot welded to a metal lead frame. The entire circuit is then encapsulated by an injection molded plastic technique.

For some critical integrated circuits, several types of ceramic packages are used. These are capable of withstanding higher temperatures and are generally hermetically sealed to provide an extra clean and safe environment for the circuit.

The Digital System

It usually requires several digital ICs to implement a given digital circuit design. Once the proper ICs are chosen, they must be "layed-out" on **printed circuit boards, or PCs**. Several ICs can be placed on a single PC, depending on its

Self-Test Review

12. List three digital design considerations.

A. ____________

B. ____________

C. ____________

13. Define propagation delay, or t_p.

____

____

14. Rise and fall time of a digital signal is measured between the ________ and ________ points of the rising or falling signal.

15. A digital IC specification gives the following information:

$t_p = 10$ nS

power dissipation = .015 watts

Calculate the speed-power product.

16. Digital IC devices used in this course are the result of ________ scale and ________ scale integration.

17. The three basic methods of packaging a digital IC are:

A. ____________

B. ____________

C. ____________

18. A TTL gate has a fanout of 10. To increase this level you must:

____

____

19. List the four major design levels of a digital system.

A. ____________

B. ____________

C. ____________

D. ____________

DIGITAL IC TECHNOLOGY

Since you will soon be designing and experimenting with real digital devices, it is time to acquaint you with the characteristics of the various types of IC's that are commercially available. You will first learn about the general characteristics of different types of digital IC's. Then, you will be aquainted with several popular digital IC families.

The term **monolithic** is derived from Greek and means "one-stone." Consequently, a monolithic IC is one that is constructed entirely on one-stone, or single chip, of silicon. Semiconductor materials are diffused into the basic silicon substrate to form the various junctions making up components such as diodes, transistors, capacitors, and resistors. The semiconductor materials to be diffused into the silicon substrate are in a gaseous form and are deposited on the substrate through a series of masking operations under very high temperature. The resulting single-chip IC contains all the components and interconnections required to perform a given digital operation. Most digital IC's are constructed using the monolithic technique.

From Figure 1-29, you can see that there are basically two types of digital ICs: **bipolar** and **MOS**. The difference between these two technologies is primarily due to the types of transistors that are used to construct the circuits. Each technology has its own advantages and disadvantages as related to the characteristics discussed earlier. Most of the SSI and MSI devices are constructed using bipolar technology since it is faster than MOS. On the other hand, most of the LSI and VLSI devices are MOS devices, since MOS has lower power dissipation and higher packing density than bipolar.

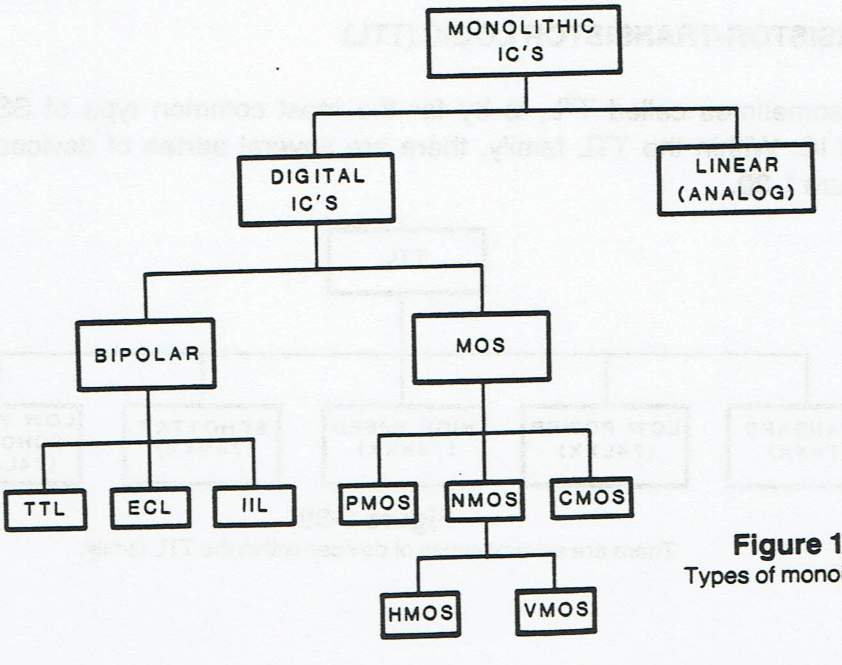


Figure 1-29
Types of monolithic ICs.

In general, all TTL series devices operate using a single 5-volt supply. Ideally, a logic 0 is represented by a 0-volt, or ground, level and a logic 1 is represented by a 5-volt level. However, in the real world there are tolerances that must be considered. As a result, there is a band of voltage levels that define both a logic 0 and logic 1 level.

When looking at a TTL device data sheet you might encounter the following terms:

V_{OL} = 0.4 volts = the maximum output voltage that can be a logic 0.
V_{OH} = 2.4 volts = the minimum output voltage that can be a logic 1.

In other words, a logic 0 TTL **output** can range from 0 to 0.4 volts, while a logic 1 output can be anywhere between 2.4 and 5 volts.

Notice the meaning of the subscripts in the above designations, The "O" stands for output in both. The "L" stands for low, and the "H" means high.

Both of the above specifications are related to the output of a TTL device. TTL device inputs have a different range of voltages that are used to define logic levels.

V_{IL} = 0.8 volts = the maximum input voltage that can be a logic 0.
V_{IH} = 2.0 volts = the minimum input voltage that can be a logic 1.

This says that an input anywhere between 0 and 0.8 volts is recognized as a logic 0, while an input between 2 and 5 volts is recognized as a logic 1. Note that the "I" subscript stands for input in the above voltage specifications.

Standard TTL (7400 Series)

The standard TTL series of devices was introduced by Texas Instruments in 1964. This series is popularly known as the **7400** series of devices. The 7400 series of standard TTL devices continues to be one of the most popular and widely used families of digital ICs. Its popularity is primarily the result of its extremely low cost and the availability of a wide range of SSI and MSI devices. Several common 7400 series ICs are listed in Table 1-1.

Table 1-1 Common 7400 Series ICs

Device Number	Functional Description
7400	Quad 2-bit NAND
7402	Quad 2-bit NOR
7404	Hex Inverter
7408	Quad 2-bit AND
7410	Triple 3-bit NAND
7427	Triple 3-bit NOR
7430	8-bit NAND
7486	Quad 2-bit XOR

You can determine the contents and logic operation of a given IC from its functional description. For example, a 7400 IC contains four (quad) 2-bit NAND gates, a 7404 contains six (hex) inverters, and a 7430 contains a single 8-bit NAND gate.

All of the ICs listed in Table 1-1 are SSI devices. The series includes almost 50 such devices, in addition to more than 100 MSI devices. You will be acquainted with the 7400 series of MSI devices in Units 2 and 3.

Some typical characteristics of standard TTL devices are listed in Table 1-2.

Table 1-2 Standard (7400) TTL Characteristics

Propagation Delay:	10 nanoseconds (nS)
Power Dissipation:	10 milliwatts (mw)
Fanout:	10
Noise Immunity:	High (0.4 volt margin typical)
Supply Voltage:	V_{CC} = +5 volts (± 10 percent)

High Speed TTL (74H Series)

In high speed TTL devices, the internal resistances are decreased to lower the internal RC time constant. Consequently, the propagation delay time is decreased, resulting in higher speed operation. The obvious trade-off is higher power consumption. High speed TTL devices are numbered 74H00, 74H02, 74H04, and so on. Table 1-4 lists the characteristics of these devices.

Table 1-4: High Speed (74H) TTL Characteristics

Propagation Delay:	6 nanoseconds (nS)
Power Dissipation:	22 milliwatts (mw)
Fanout:	10
Noise Immunity:	High (0.4 volt margin typical)
Supply Voltage:	V_{CC} = +5 volts (± 10 percent)

Schottky TTL (74S Series)

In Schottky TTL devices, a special diode, called a **Schottky diode**, is added between the collector and base of each internal switching transistor. The purpose of the diode is to increase the switching speed of the transistors, resulting in a lower propagation delay for for the circuit. A Schottky TTL device is about twice as fast as a high speed TTL device and has lower power consumption. These devices are numbered 74S00, 74S02, 74S04, etc. The characteristics of this series are listed in Table 1-5.

Table 1-5: Schottky TTL (74S) Characteristics

Propagation Delay:	3 nanoseconds (nS)
Power Dissipation:	19 milliwatts (mw)
Fanout:	10
Noise Immunity:	High (0.4 volt margin typical)
Supply Voltage:	V_{CC} = +5 volts (± 10 percent)

One of the major advantages of an ECL gate is the availability of both the normal and inverted outputs simultaneously. Notice from Figure 1-32 that the 2-bit gate shown provides both an OR and a NOR function, without the use of an external inverter. In addition, the output of one ECL gate can be connected directly to the output of another ECL gate to produce a wire-ORed operation. This is not possible with TTL.

Notice also from Figure 1-32 that an external resistor, R_p, must be connected between the gate output and the supply voltage source, V_{EE}. This resistor is called a **pull-down resistor** and is required on all the gate outputs. The pull-down resistor value can be anywhere from 270 ohms to 2000 ohms (2 kΩ), depending on the power and load requirements of the circuit.

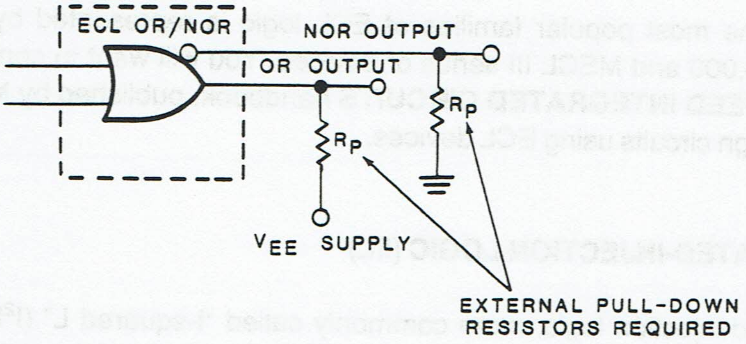


Figure 1-32

A typical ECL device has complementary outputs, but requires pull-down resistors.

The general characteristics of ECL are listed in Table 1-7. As you can see, ECL is extremely fast, but suffers from high power dissipation. In addition, notice that ECL requires a −5.2 volt supply. As a result, a logic 0 is defined as a −1.75 volt level, and a logic 1 is defined as −0.9 volts. Furthermore, notice that the noise immunity level is 0.2 volts for ECL compared with 0.4 volts for TTL.

MOS Technology

The acronym MOS stands for metal oxide semiconductor. The name comes from the kind of transistors that are integrated into MOS ICs. These transistors are called **MOSFETs**, which stands for metal oxide semiconductor field-effect transistors. In general, MOS technology provides greater packing densities and consumes less power the bipolar.

MOSFETs come in three varieties: P-channel MOSFETs called **PMOS**, N-channel MOSFETs called **NMOS**, and complimentary MOSFETs called **CMOS**. Integrated circuits using these different varieties of MOSFETs are thus called PMOS, NMOS, and CMOS devices.

PMOS

This is the oldest of the MOS technologies and is mostly obsolete. PMOS ICs were originally developed for the microprocessor and semiconductor memory markets. Although it is much slower than bipolar, it has higher density and consumes much less power, thereby making it attractive to these LSI markets.

NMOS

It wasn't long before NMOS began replacing PMOS in the LSI markets. The reason is that NMOS has higher density and is faster than PMOS. Originally, NMOS was harder to manufacture and thus more expensive than PMOS. However, as processing technology improved, it became easier and cheaper to make NMOS devices. NMOS is now the dominant technology in the LSI and VLSI fields, which includes mostly microprocessors and semiconductor memories.

Several "spin-offs" have resulted from NMOS technology. Two of the more popular are **HMOS** and **VMOS**. The abbreviation HMOS stands for high density MOS. It is actually NMOS technology, but uses special processing techniques that result in chips with higher density and less power consumption. Many VLSI devices are manufactured using HMOS technology.

The abbreviation VMOS stands for vertical MOS. This technology uses a V-shaped groove in the silicon chip to achieve greater packing densities. Its characteristics are similar to NMOS. However, it's not as widely used since the manufacturing processes are not well developed.

Let's examine the above characteristics a bit closer. First, notice that the power dissipation is a linear function of both the supply voltage and operating frequency. This means that if the supply voltage was increased to 10 volts, a CMOS device would have a static power dissipation of about 5 nw and dissipate approximately 2 mw at a switching speed of 1 MHz. Likewise, if the switching speed were reduced to 500 KHz, the operating power dissipation would reduce to about 0.5 mw using a 5-volt supply.

Notice also that the noise immunity is a function of the supply voltage. If the supply voltage were 12 volts, the noise margin would be 45% of 12 volts, or 4.5 volts!

Another extreme advantage of CMOS is its ability to operate on a supply voltage anywhere between 3 and 18 volts. This feature, coupled with low power dissipation, makes CMOS an ideal choice for battery supply applications. Notice that a logic 0 is always defined as a 0 volt, or ground potential. A logic 1 is defined to be the supply voltage, or V_{DD}, potential. Many CMOS circuits are designed to operate from a standard 9-volt transistor radio battery.

CMOS IC's are numbered using a 74C or 74HC number. The 74C stands for standard CMOS and the 74HC stands for high speed CMOS. As you can see from Table 1-9, a CMOS IC with a 74C or 74HC number has the same function as its 7400 series TTL counterpart. In fact, the respective CMOS and TTL ICs are pin-for-pin compatible.

Table 1-9: Some Common 74C and 74HC series CMOS ICs

Device Number	**Functional Description**
74C00 or 74HC00	Quad 2-bit NAND
74C02 or 74HC02	Quad 2-bit NOR
74C04 or 74HC04	Hex Inverter
74C10 or 74HC10	Triple 3-bit NAND
74C27 or 74HC27	Triple 3-bit NOR
74C30 or 74HC30	Single 8-bit AND
74C86 or 74HC86	Quad 2-bit XOR

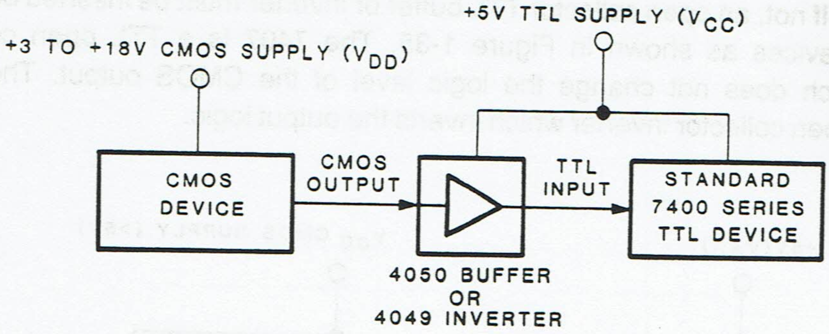

+3 TO +18V CMOS SUPPLY (V_{DD})
+5V TTL SUPPLY (V_{CC})
CMOS DEVICE
CMOS OUTPUT
TTL INPUT
STANDARD 7400 SERIES TTL DEVICE
4050 BUFFER OR 4049 INVERTER

Figure 1-33
A 4050 buffer or 4049 inverter must be used between a CMOS output and a standard 7400 series TTL input.

With this interface, the CMOS device can operate on any supply voltage, but the 4050 buffer or 4049 inverter must operate on the 5-volt TTL supply. Thus, the input of a TTL device can be connected to the output of a CMOS device operating on any supply voltage.

Connecting TTL to CMOS

When connecting a TTL output to a CMOS input, a **pull-up** resistor must be used as shown in Figure 1-34. This is because a logic 1 TTL output can be as low as 2.4 volts, while a CMOS logic 1 must be at least 3 volts. The pull-up resistor "pulls" the TTL output up to 5 volts. The pull-up resistor value must be between 1 kΩ and 10 kΩ.

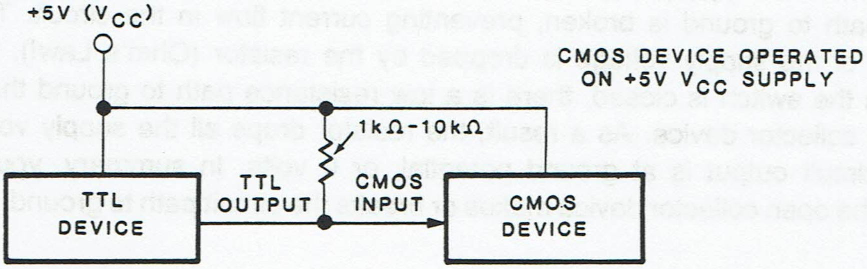


Figure 1-34
A pull-up resistor must be used on the TTL output to drive a CMOS input. This only works if the CMOS device is operating on a 5-volt supply.

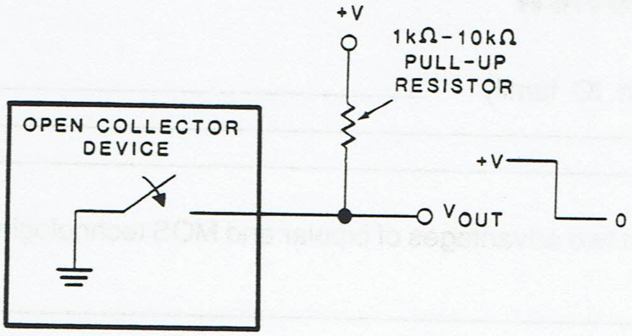


Figure 1-36
Open collector devices act like a switch in a pull-up resistor circuit.

All open collector devices require that a separate **external** pull-up resistor be connected to the output of each gate within the device. If this is not done, the device will not operate. The pull-up resistor value should be between 1 kΩ and 10 kΩ. The other side of the pull-up resistor is then connected to the CMOS supply as shown in Figure 1-35 so that the TTL output is pulled-up to the CMOS logic level. Notice that power for the open collector device is supplied by the +5-volt (TTL) supply line. This is necessary since it is a TTL device.

PRECAUTIONS WHEN WORKING WITH MOS DEVICES

Any MOS device can be damaged by static electricity like that which can build up on your clothes. Although many MOS devices are internally protected against static electricity, the following precautions should be taken when working with these ICs:

1. Avoid touching the IC pins and make sure you are grounded. This can be accomplished by wearing a conductive wrist strap that is grounded.
2. Avoid wearing nylon clothing when working with MOS ICs.
3. Never insert or remove MOS ICs with the circuit power on.
4. When not using a MOS device, store it in conductive foam or an anti-static tube.
5. When soldering to the MOS IC pins, make sure the tip of the soldering iron is grounded.

25. A 74C00 Quad 2-bit NAND chip is pin-for-pin compatible with a 4011 Quad 2-bit NAND device. ____________
True/False

26. How must standard TTL devices be connected to CMOS devices if both are used in a digital circuit? ____________________________
____
____
____

The solution to questions 23 and 24 follow:

23A. Low level noise immunity:

$$\begin{aligned} V_{NL} &= V_{IL} - V_{OL} \\ &= 1.2\text{ V} - 0.7\text{ V} \\ &= 0.5\text{ V} \end{aligned}$$

23B. High level noise immunity:

$$\begin{aligned} V_{NH} &= V_{OH} - V_{IH} \\ &= 2.0\text{ V} - 0.7\text{ V} \\ &= 1.3\text{ V} \end{aligned}$$

24A. Table 1-8 shows the static operating characteristics of CMOS to be 2.5 nw at 5 volts. Since you wish to determine the static power at 12 volts, divide 12 volts by 5 volts. This gives a ratio of 2.4. Then multiply the static power at 5 volts by the voltage ratio.

$$\begin{aligned} \text{Static Power} &= \frac{12\text{ V}}{5\text{ V}} \times 2.5\text{ nw} \\ &= 6\text{ nw} \end{aligned}$$

24B. Table 1-8 shows the dynamic operating characteristics of CMOS to be 1 mw at 1 MHz with a 5-volt supply. Since you wish to determine the dynamic power at 12 volts, divide 12 volts by 5 volts. This gives a ratio of 2.4. Because the frequency is still 1 MHz, you multiply the dynamic power at 5 volts by the voltage ratio.

$$\begin{aligned} \text{Dynamic Power} &= \frac{12\text{ V}}{5\text{ V}} \times 1\text{ mw} \\ &= 2.4\text{ mw} \end{aligned}$$

UNIT EXAMINATION

1. The truth table in Figure 1-37 represents a:
   - A. 2-bit XNOR gate.
   - B. 3-bit NAND gate.
   - C. 3-bit XOR gate.
   - D. 3-bit XNOR gate.

A	B	C	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Figure 1-37
Truth Table for Question 1.

2. A general mathematical formula that can be used to determine the number of input possibilities for an n-bit gate is:
   - A. $n + 2$
   - B. $2^n + 1$
   - C. 2^n
   - D. $2^n - 1$

3. The Boolean expression for the circuit in Figure 1-38 is:
   - A. $y = AB$
   - B. $y = A + B$
   - C. $y = \overline{AB}$
   - D. $y = \overline{A + B}$

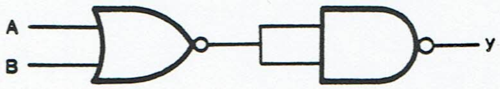


Figure 1-38
Circuit for Question 3.

6. If $A_3A_2A_1A_0$ = 0101 and $\overline{\text{COMPLEMENT}}$ = 0 in Figure 1-41, then $y_3y_2y_1y_0$ is:
   - A. 0000
   - B. 0101
   - C. 1010
   - D. 1111

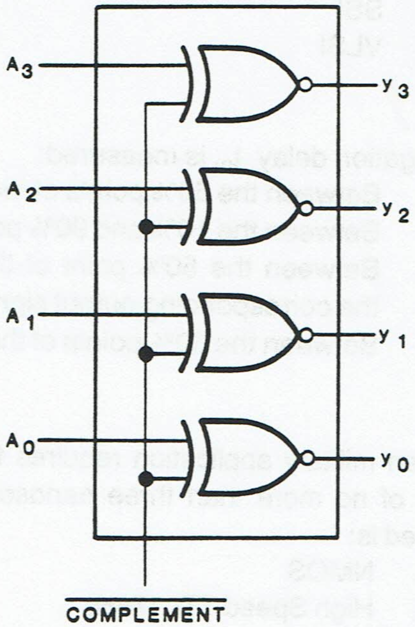


Figure 1-41
Circuit for Question 6.

7. The speed-power product of a digital IC is found to be 35. If the power dissipation of the IC is 5 mw/gate, the propagation delay is:
   - A. .125 nanoseconds
   - B. 7 nanowatts
   - C. 7×10^{-9} seconds
   - D. 7 milliseconds

13. A CMOS device is to be operated at 750 KHz using a 5-volt supply. Its dynamic, or operating, power dissipation is:
    - A. 750 microwatts
    - B. 0.75 watts
    - C. 1 milliwatt
    - D. 0.5 milliwatts

14. The CMOS device in question 13 has a noise immunity level of:
    - A. 0.4 volts
    - B. 2.25 volts
    - C. 4.5 volts
    - D. 0.225 volts

15. A digital IC with a 74HC number is a:
    - A. High speed TTL device.
    - B. ECL device.
    - C. CMOS device.
    - D. None of these.

16. Connecting a CMOS output to a 74LS input requires:
    - A. A 4050 buffer.
    - B. A pull-up resistor.
    - C. A 7407 open collector buffer.
    - D. Nothing.

17. Connecting a 74LS output to a 9 volt CMOS input requires:
    - A. A 4050 buffer.
    - B. A pull-up resistor.
    - C. A 7407 open collector buffer.
    - D. Nothing.

EXAMINATION ANSWERS

1. D — The truth table in Figure 1-37 represents a 3-bit XNOR gate.

2. C — A general mathematical formula that can be used to determine the number of input possibilities for an n-bit gate is 2^n.

3. B — The Boolean expression for the circuit in Figure 1-38 is $y = A + B$.

4. C — If $R = 0$ and $S = 1$ in Figure 1-39, the Q and $\overline{Q}$ outputs must be $Q = 1$ and $\overline{Q} = 0$.

5. D — The timing diagram in Figure 1-40 is for a XNOR gate.

6. C — If $A_3A_2A_1A_0 = 0101$ and $\overline{\text{COMPLEMENT}} = 0$ in Figure 1-41, then $y_3y_2y_1y_0$ is 1010.

7. C — The speed-power product of a digital IC is found to be 35. If the power dissipation of the IC is 5 mw/gate, the propagation delay is 7×10^{-9} seconds.

8. A — Microprocessors are the direct result of LSI.

9. C — Propagation delay, t_p, is measured between the 50% point of the input signal and the 50% point of the corresponding output signal.

10. C — A given military application requires the digital ICs have a propagation delay of no more than three nanoseconds. The IC technology that must be used is ECL.

11. D — Power dissipation is directly proportional to switching speed in CMOS.

12. D — A digital IC circuit requires a supply voltage of from +3 to +18 volts. The type of ICs that this circuit must be using is CMOS.

13. A — A CMOS device is to be operated at 750 KHz using a 5-volt supply. Its dynamic, or operating, power dissipation is 750 microwatts.

14. B — The CMOS device in question 13 has a noise immunity level of 2.25 volts.

15. C — A digital IC with a 74HC number is a CMOS device.

16. D — Nothing is required when connecting a CMOS output to a 74LS input.

17. C — Connecting a 74LS output to a 9 volt CMOS input requires a 7407 open collector buffer.

18. A — The NAND gate in Figure 1-42 must be a standard TTL device.

19. A — The supply voltage for the NOR gate in Figure 1-42 must be +5 volts.

20. C — Given the Boolean expression $y = \overline{A}\overline{B} + BC$, the equivalent digital circuit would be a 2-bit bubbled AND gate and a 2-bit AND gate driving a 2-bit OR gate. Note: By DeMorgan's theorem, the bubbled AND gate could be replaced with a 2-bit NOR gate.

18. The NAND gate in Figure 1-42 must be a:
    - A. Standard TTL device.
    - B. Low power Schottky TTL device.
    - C. CMOS device.
    - D. High speed CMOS device.

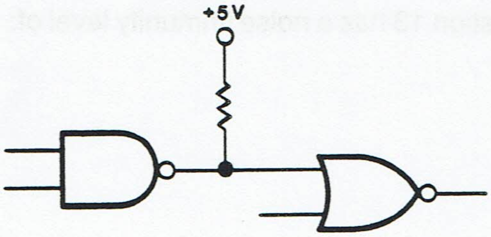


Figure 1-42
Circuit for Questions 18 and 19.

19. The supply voltage for the NOR gate in Figure 1-42 must be:
    - A. +5 volts.
    - B. Anywhere from +3 volts to +18 volts.
    - C. Greater than +5 volts.
    - D. Less than +5 volts.

20. Given the Boolean expression $y = \overline{A}\overline{B} + BC$, the equivalent digital circuit would be:
    - A. Two 2-bit OR gates driving a 2-bit AND gate.
    - B. Two 2-bit AND gates driving an OR gate.
    - C. A 2-bit bubbled AND gate and a 2-bit AND gate driving a 2-bit OR gate.
    - D. A 2-bit bubbled AND gate and a 2-bit AND gate driving a 4-bit OR gate.

8. Microprocessors are the direct result of which digital integration level?
   - A. LSI
   - B. MSI
   - C. SSI
   - D. VLSI

9. Propagation delay, t_p, is measured:
   - A. Between the 50% points of the input signals.
   - B. Between the 10% and 90% points of the signal.
   - C. Between the 50% point of the input signal and the 50% point of the corresponding output signal.
   - D. Between the 50% points of the output signals.

10. A given military application requires the digital ICs to have a propagation delay of no more than three nanoseconds. The IC technology that must be used is:
    - A. NMOS
    - B. High Speed TTL (74H)
    - C. ECL
    - D. CMOS

11. Power dissipation is directly proportional to switching speed in:
    - A. TTL
    - B. ECL
    - C. VMOS
    - D. CMOS

12. A digital IC circuit requires a supply voltage of from +3 to +18 volts. This circuit must be using what type of ICs?
    - A. TTL
    - B. ECL
    - C. HMOS
    - D. CMOS

4. Suppose R = 0 and S = 1 in Figure 1-39. The Q and $\overline{Q}$ outputs must be:

   A. $Q = 0, \overline{Q} = 0$
   B. $Q = 0, \overline{Q} = 1$
   C. $Q = 1, \overline{Q} = 0$
   D. $Q = 1, \overline{Q} = 1$

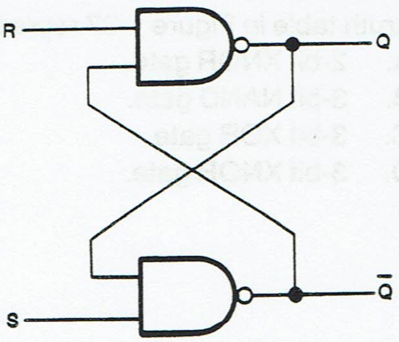


Figure 1-39
Circuit for Question 4.

5. The timing diagram in Figure 1-40 is for a __________ gate.

   A. NAND
   B. NOR
   C. XOR
   D. XNOR

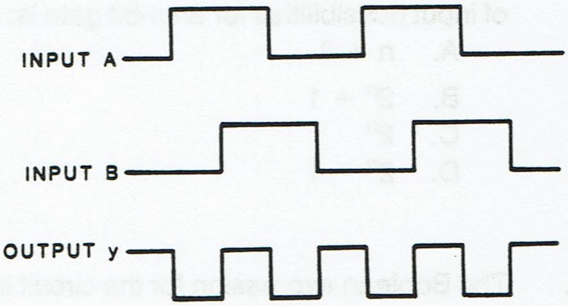


Figure 1-40
Timing Diagram for Question 5.

SUMMARY

Analog signals are continuous and assume an infinite number of values between some minimum and maximum value. Digital signals only assume two levels, 1 or 0. As a result, the binary number system and Boolean algebra are used to represent digital signals and circuits.

Digital logic gates are devices that operate on one or more digital input signals to produce a digital output. The seven fundamental gates of all digital electronics are the inverter, OR, NOR, AND, NAND, XOR, and XNOR. These logic gate operations can be described by a symbol, truth table, or Boolean expression.

The major characteristics of digital devices that must be considered in order to design working digital circuits are speed, power dissipation, fanout, noise immunity, density, packaging, and cost. Each of the major digital IC families such as TTL, ECL, NMOS, and CMOS have advantages and disadvantages as related to these design considerations. It is the job of the digital designer to determine the IC technology used to implement the design. The application will dictate the relative importance of the above design considerations, thereby allowing the designer to choose the appropriate IC technology.

No special circuits are required when connecting devices within a given IC family. However, combining families within a design may require special interface circuits. The most common combination is TTL and CMOS. Connections between these two families require buffering and/or pull-up resistor circuits.

Special precautions must be taken when handling MOS devices since they can be damaged by static electricity. At a minimum, you should ground yourself through a ground strap and avoid touching the pins of a MOS device.

Before you take the Unit Examination, perform Experiments 1 through 4, in Unit 8.

Answers

20. A family is a group of ICs that operate on the same supply voltage and use the same voltages to represent logic levels.

21A. Bipolar — High Speed

21B. Bipolar — Low Cost for SSI and MSI devices.

21C. Biploar — Availability for SSI and MSI devices.

21D. MOS — High Density

21E. MOS — Low Power Dissipation

21F. MOS — Low Cost for SSI and MSI devices.

21G. MOS — High Fanout (CMOS)

21H. MOS — High Noise Immunity (CMOS)

22. CMOS — Because of its very low power dissipation.

23A. Low Level = 0.5 V

23B. High Level = 1.3 V

24A. Static Power = 6 nw

24B. Dynamic Power = 2.4 mw

25. False — 4000 series CMOS devices are not pin-for-pin compatible with 74C or 74HC series CMOS devices.

26. A CMOS output must be buffered using a CMOS buffer if it is connected to a TTL input (see Figure 1-33). Low power Schottky TTL does not require such buffering.

A pull-up resistor must be used on a TTL output to drive a CMOS input if the CMOS device is operated on a 5-volt supply (see Figure 1-34). If the CMOS supply is above 5 volts, an open collector buffer must be used between the TTL output and CMOS input (see Figure 1-35).

Self-Test Review

20. What is an IC family? ______________________________
____
____

21. List at least two advantages of bipolar and MOS technologies.

Bipolar ______________________________
Bipolar ______________________________
MOS ______________________________
MOS ______________________________

22. A digital circuit designed for a satellite application would most likely use ______ ICs.

23. A digital IC specification provides the following information:

$V_{IL} = 1.2V$
$V_{IH} = 0.7V$
$V_{OH} = 2.0V$
$V_{OL} = 0.7V$

Calculate the low level and high level noise immunity for this IC.

A. Low Level = ________
B. High Level = ________

24. A CMOS device is to be operated at 1 MHz using a 12V supply. Calculate its approximate static and dynamic (operating) power dissipation.

A. Static Power = __________
B. Dynamic Power = __________

The pull-up resistor technique works fine as long as the CMOS device is operating on 5 volts. If not, an open collector TTL buffer or inverter must be inserted between the two devices as shown in Figure 1-35. The 7407 is a TTL open collector buffer which does not change the logic level of the CMOS output. The 7406 is a TTL open collector inverter which inverts the output logic.

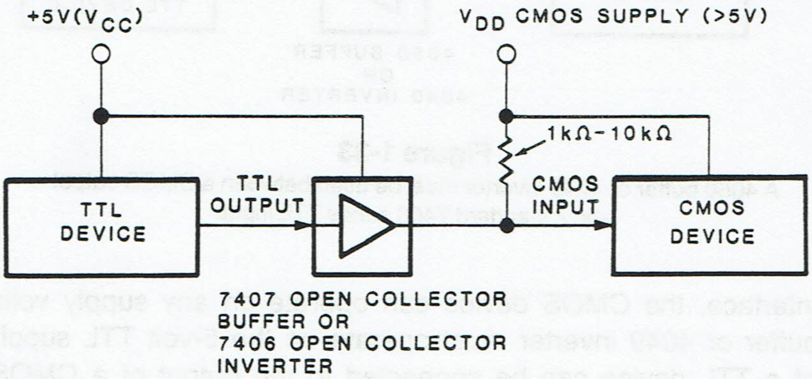


Figure 1-35

A 7407 open collector buffer or 7406 open collector inverter must be used between a TTL output and CMOS input if the CMOS device is operated using a supply above +5 volts.

An open collector device acts like a switch in a pull-up resistor circuit. Look at Figure 1-36. When the switch is open, there is an infinite resistance path to ground and the circuit output is pulled to the +V supply potential by the pull-up resistor. The supply potential is seen at the bottom of the pull-up resistor because the path to ground is broken, preventing current flow in the circuit. Therefore, none of the supply voltage is dropped by the resistor (Ohm's Law!). However, when the switch is closed, there is a low resistance path to ground through the open collector device. As a result, the resistor drops all the supply voltage and the circuit output is at ground potential, or 0 volts. In summary, you can say that the open collector device makes or breaks the circuit path to ground.

In addition to the 74C and 74HC series, there is also a 4000 series of CMOS devices. This was the original series of CMOS devices and is still readily available. However, there is absolutely no correlation between the numbering of the 4000 series and 74C series of devices. Thus, you cannot substitute a 4000 series IC directly for a 74C or 74HC series IC. Several common 4000 series ICs are listed in Table 1-10.

Table 1-10: Some Common 4000 Series CMOS ICs

Device Number	Functional Description
4000	Dual 3-bit NOR plus Inverter
4001	Quad 2-bit NOR
4002	Dual 4-bit NOR
4011	Quad 2-bit NAND
4049	Hex Inverter
4050	Hex Buffer
4068	Single 8-bit NAND
4081	Quad 2-bit AND

Connecting CMOS to TTL

There are times when CMOS and TTL ICs will be connected together. A CMOS output can be connected directly to the input of a 74LS TTL device, provided that the CMOS device is operating on a 5-volt supply. However, a CMOS output cannot be connected to the input of a standard 7400 series device. The interface between the two devices must be **buffered** as shown in Figure 1-33. Here, a 4050 CMOS **buffer** has been inserted between the CMOS output and TTL input. A buffer increases the signal power of the output but does not change its logic level. An inverter, such as a 4049, will also do the job but the logic level is inverted.

CMOS

We will concentrate on CMOS in this course since, along with TTL, it is the dominant technology used in SSI and MSI digital design. The reason for this is because of its characteristic high packing density and very low power consumption.

CMOS is actually a combination of PMOS and NMOS technology. The unique feature of CMOS is that virtually no power is dissipated when a gate is in a logic 0 or a logic 1 state. A constant logic 0 or 1 state is called the **static state**. The only time a CMOS gate dissipates any appreciable power is when it switches from one logic state to another. In fact, the amount of power dissipation is directly proportional to the switching speed of the gate.

The characteristics of CMOS are listed in Table 1-8. If a digital designer were to specify the perfect logic circuit, it would be CMOS for most applications. CMOS logic circuits offer a balanced combination of characteristics that make them highly versatile and desirable. Notice from Table 1-8 that CMOS features moderately high speed operation, very low power dissipation, high fanout, excellent noise immunity, and a wide range of power supply options.

Table 1-8: CMOS Characteristics

Propagation Delay:	10 to 200 nanoseconds (nS)
Power Dissipation:	2.5 nanowatts (nw) static with 5 V supply
	1 milliwatt (mw) at 1 Megahertz (MHz) with 5 V supply
Fanout:	50 +
Noise Immunity:	Very High (45 % of supply voltage)
Supply Voltage:	V_{DD} = +3 volts to +18 volts
	Logic 0 = 0 volts
	Logic 1 = V_{DD}

Table 1-7: Emitter-Coupled Logic Characteristics

Propagation Delay:	1 to 2.5 nanoseconds (nS)
Power Dissipation:	25 to 60 milliwatts (mw)
Fanout:	10 to 25
Noise Immunity:	Low (0.2 volt margin typical)
Supply Voltage:	V_{EE} = − 5.2 volts
	Logic 0 = − 1.75 volts
	Logic 1 = − 0.9 volts

One of the most popular families of ECL logic is represented by the Motorola MECL 10,000 and MECL III series of devices. You will want to consult the **MECL HIGH-SPEED INTEGRATED CIRCUITS** handbook, published by Motorola, if you ever design circuits using ECL devices.

INTEGRATED-INJECTION LOGIC (IIL)

Integrated-injection logic, more commonly called "I-squared L" (I^2L), is a relative newcomer to the list of bipolar logic families. It was developed for high density applications where the characteristic high speed of bipolar is required. In particular, I^2L was developed to compete with MOS technology in microprocessor applications such as games, calculators, and watches.

One reason that I^2L is more dense than other bipolar families is that no resistors are required in the internal circuit, thereby reducing the size of each gate within the IC. This also reduces the power consumption of the device. Typical power consumption is 5 milliwatts per gate with a propagation delay of 5 nanoseconds.

Because of its high density, high speed, and low power consumption, I^2L promises to have a major impact on the LSI field. However, more development needs to be done before all the predictions for I^2L are realized. The cost of I^2L is still relatively high compared to the various MOS technologies.

Low Power Schottky TTL (74LS Series)

Low power Schottky combines the features of low power TTL and Schottky TTL to produce a device that has both low power dissipation and high speed. With low power Schottky ICs, the internal resistances are increased to lower the power consumption and Schottky diodes are added to decrease the transistor switching times. These devices are numbered 74LS00, 74LS02, 74LS04, and so forth. The characteristics of the 74LS series are listed in Table 1-6.

Table 1-6: Low Power Schottky (74LS) Characteristics

Propagation Delay:	5 nanoseconds (nS)
Power Dissipation:	2 milliwatts (mw)
Fanout:	20
Noise Immunity:	High (0.3 volt margin typical)
Supply Voltage:	V_{CC} = +5 volts (± 10 percent)

Of the five different series of TTL devices, standard TTL and low power Schottky TTL are by far the most common in commercial digital circuits. In fact, the cost is about the same for 74 and 74LS devices, with 74LS devices developing a slight cost advantage in recent years. The cost of 74L, 74H, and 74S devices is much higher by comparison. This is one of the reasons that most digital designers will specify either standard TTL or low power Schottky TTL. You might have already noticed that most of your experiment parts are 74LS series devices.

EMITTER-COUPLED LOGIC (ECL)

Emitter-coupled logic is the fastest digital logic available. It achieves its speed from the way that the internal switching transistors are connected. Because the internal circuitry is more complex, ECL devices are higher cost and dissipate more power than other bipolar devices. Typical applications of ECL devices include real time process control systems, signal processors, navigation systems, and just about any application that requires very high speed performance.

In addition to these characteristics, you should be aware that the 7400 series of devices is designed to operate over a temperature range of 0° to 70° C (Celsius). This range is more than adequate for most commercial applications. However, some applications dictate a greater operating temperature range. Many military applications fall into this category. For this reason, a series of TTL devices called the 5400 series is available with an operating temperature range of −55° to 125° C. The 5400 series is a direct replacement for the 7400 series. In other words, a 5404 inverter can be substituted directly for a 7404 inverter. They are said to be "pin-for-pin" compatible. Although 5400 series devices operate over a wide temperature range, the trade-off is cost. They are much more expensive than 7400 series devices.

Low Power TTL (74L Series)

Low power TTL devices are similar to standard TTL and have numbers of 74L00, 74L02, 74L04, and so on. The only difference is that the internal resistor values are approximately ten times higher, resulting in one-tenth the power consumption of the standard TTL IC. However, speed is sacrificed for the lower power consumption. Increasing the internal resistances of the device increases the propagation delay time due to higher RC time constants. Typical characteristics for the 74L series are listed in Table 1-3.

Table 1-3: Low Power (74L) TTL Characteristics

Propagation Delay:	**35 nanoseconds (nS)**
Power Dissipation:	**1 milliwatt (mw)**
Fanout:	**20**
Noise Immunity:	**High (0.4 volt margin typical)**
Supply Voltage:	**V_{CC} = +5 volts (± 10 percent)**

The above input and output voltage bands are shown in Figure 1-31. Observe that there is a difference between the input bands and the output bands. The difference between the two sets of bands is called the **noise margin**.

For a logic 0 the noise margin is called the **low level noise margin**, or $\mathbf{V_{NL}}$, and is calculated as follows:

$$
\begin{aligned}
V_{NL} &= V_{IL} - V_{OL} \\
&= 0.8\ \text{V} - 0.4\ \text{V} \\
&= 0.4\ \text{volts}
\end{aligned}
$$

Likewise, for a logic 1 the noise margin is called the **high level noise margin,** or $\mathbf{V_{NH}}$, and is calculated as follows:

$$
\begin{aligned}
V_{NH} &= V_{OH} - V_{IH} \\
&= 2.4\ \text{V} - 2.0\ \text{V} \\
&= 0.4\ \text{volts}
\end{aligned}
$$

As you can see, a TTL device has a **low level** and **high level** noise margin of 0.4 volts. In other words, TTL devices have a 0.4 volt noise immunity level from output to input.

Finally, observe from Figure 1-31 that any input voltage between 0.8 and 2 volts is in an undefined region. This means that you cannot predict what the device will do if noise causes the input signal to wander into this region.

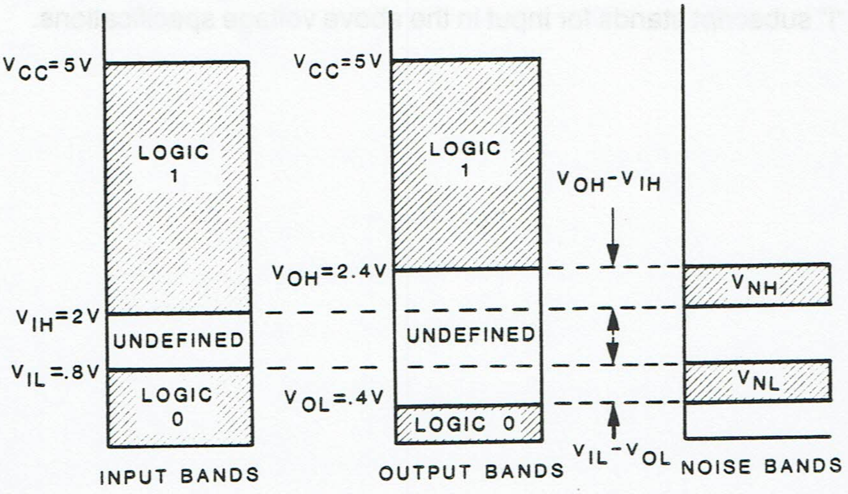


Figure 1-31
TTL logic definitions.

In addition, both bipolar and MOS technology are further subdivided into several sub-technologies as shown in the figure. These sub-technologies are called **families**. A family is a group of ICs that operate on the same supply voltage and use the same voltage levels to represent logic levels. In other words, the ICs within a given family are **compatible**. Compatibility, means that the output of one device can be connected directly to the input of another device without any special considerations. Now, let's take a closer look at each of the bipolar and MOS IC families, keeping in mind the IC characteristics discussed in the last section.

Bipolar Technology

In general, bipolar devices have advantages of being high speed and low cost for SSI and some MSI circuits. However, as the level of integration goes up, bipolar devices become more expensive as compared to their MOS counterparts. Disadvantages of bipolar devices include relatively high power dissipation, low density, and low noise immunity.

The most popular bipolar IC families in use today are:

TTL, or Transistor-Transistor Logic
ECL, or Emitter-Coupled Logic
IIL, or Integrated-Injection Logic

TRANSISTOR-TRANSISTOR LOGIC (TTL)

TTL, sometimes called T^2L, is by far the most common type of SSI and MSI digital IC. Within the TTL family, there are several **series** of devices as shown in Figure 1-30.

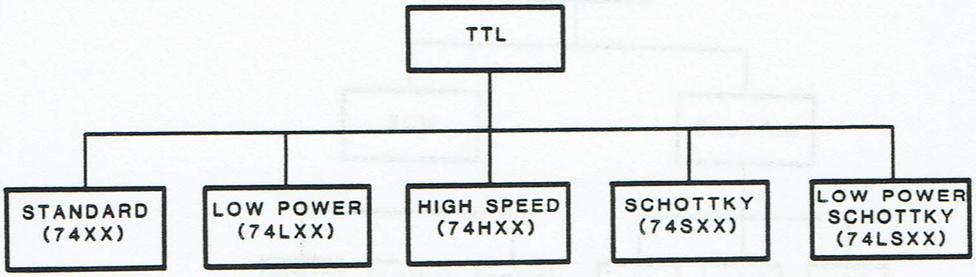


Figure 1-30
There are several series of devices within the TTL family.

Answers

12A. Speed

12B. Power Dissipation

12C. Fanout

12D. Noise Immunity

12E. Density

12F. Packaging

12G. Cost

13. Propagation delay, t_p, is the amount of time it takes the output of a digital circuit to change after its inputs have changed.

14. 10% and 90%

15. 10 nS × I5 mw = 150

16. small and medium

17A. TO5 can

17B. Flat pack

17C. Dual In-Line Package

18. Buffer the output line.

19A. Integrated Circuit

19B. Printed Circuit Board

19C. Subrack

19D. Rack

size. Although PCs have been designed to accommodate over 200 ICs, the practical limit is closer to 50 or 60 ICs. Thus, you will often see a digital system composed of multiple PC boards.

In systems where multiple PC boards are used, each board usually has a connector on one edge, called an **edge connector**. The edge connector is actually part of the PC board, but has plated contacts that terminate at the edge of the board as illustrated in Figure 1-28. The edge connectors are designed to fit into a female socket on a **mother board**, also known as a **backplane board**, which is part of a subsystem called a **subrack, or card cage**. The mother board is actually a large PC board that contains several sockets which mate with the individual PC board edge connectors. The sockets on the mother board are interconnected via plated conductor patterns on the board. Some mother boards can accommodate up to 30 PC boards.

In larger systems, the subracks are placed in a **rack**. This rack usually contains the system power supply, cooling fans, control console, and external interface connectors, in addition to the subrack(s). Still larger systems, such as mainframe computer systems, might require that the racks be bolted together to form multirack consoles.

Notice in Figure 1-28 that the digital system is a modular system, from IC to PC board to subrack to rack. It is the modular structure of a digital system that allows a designer to subdivide the overall design problem into a series of smaller, more manageable problems.

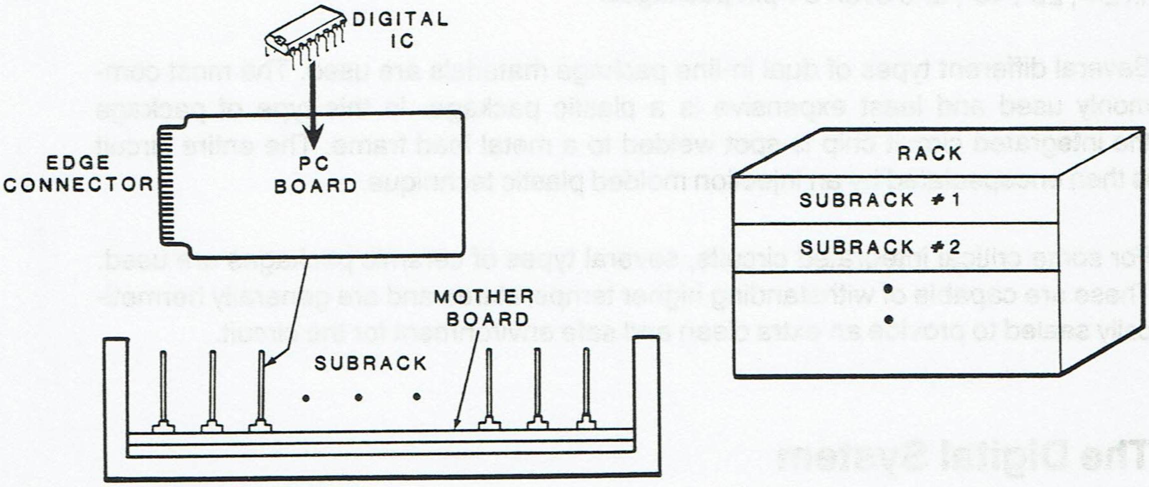


Figure 1-28
A digital system is a modular system, from IC to rack.

Although we are in the age of LSI and VLSI, many MSI, SSI, and even discrete devices are required as "glue" to support LSI and VLSI chips. In addition, many applications for digital circuits do not require such high levels of integration. The **application will always dictate** the design requirements. In this course, you will learn how to design circuits using primarily SSI and MSI devices.

Digital Circuit Packaging

A primary consideration to the integrated circuit user is the packaging and physical characteristics of integrated circuits. There are three basic methods of packaging the silicon chip. These are the **TO5** can, the **flat pack**, and the **dual in-line** (DIP) package. These three basic types of packages are illustrated in Figure 1-27.

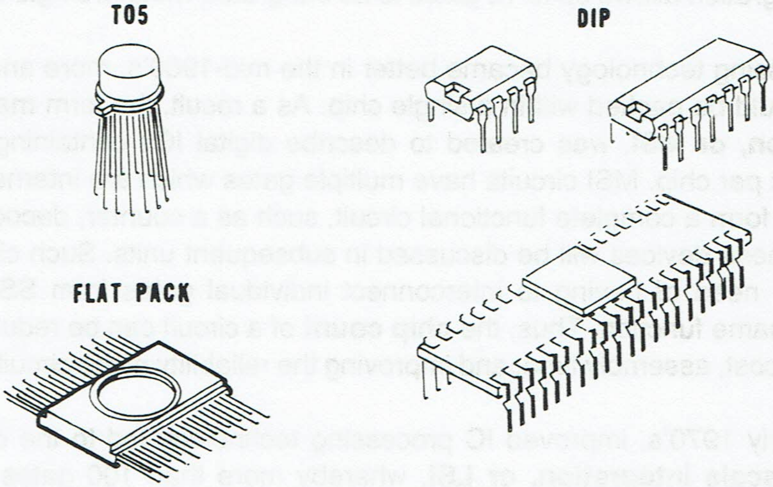


Figure 1-27
Common digital IC packaging.

TO5

The earliest form of package used for integrated circuits was the TO5 can. This is a standard configuration for packaging discrete transistors and was modified for ICs by including additional leads. This type of package is still available for some integrated circuits today, but it is not the most popular form. Its advantage is that it can dissipate a substantial amount of heat through its metal can. For that reason, the TO5 package is used mostly with linear (operational amplifiers, etc.), not digital, ICs.

Fanout

Fanout is a characteristic that indicates how much load can be connected to the output of a logic gate. A typical load is a single input to another gate of the same IC type.

Fanout is expressed in terms of the number of loads that a logic gate output can accommodate and still maintain proper operation. If a logic gate has a fanout of 10, this means that the gate output can be connected to, or **drive**, no more than 10 separate gate inputs. In other words, the output of the gate can only have a maximum of 10 parallel branches for the purpose of driving additional gate inputs. The idea of fanout is illustrated by Figure 1-26. To increase the fanout of a digital IC, its outputs must be buffered. That is, add a non-inverting gate that has a higher fanout to each of the outputs. A gate that has a high fanout is called a **buffer**.

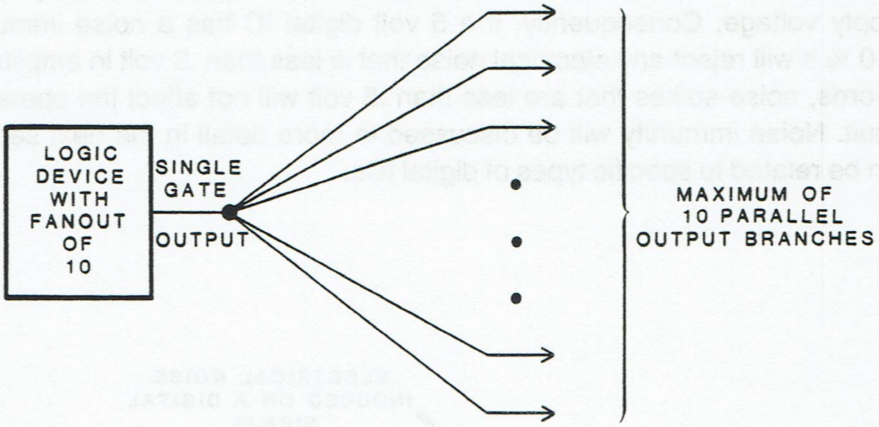

LOGIC DEVICE WITH FANOUT OF 10
SINGLE GATE
OUTPUT
MAXIMUM OF 10 PARALLEL OUTPUT BRANCHES

Figure 1-26

A single gate output from a logic device having a fanout of 10 can provide a maximum of 10 parallel output branches for the purpose of driving multiple gate inputs.

Packing Density

The term density, when related to integrated circuits, is a relative term that refers to the amount of electronic components that have been "packed," or integrated, into a single chip device. In a very high density chip, you would expect to find many more components and, therefore, much more capability than in a low density chip. Thus, packing density relates to the level of integration within the IC.

In digital circuits, power dissipation is a measure of the power consumed by the gates within a digital IC. It is an average value, since power consumption is different for logic 1 and logic 0 outputs. In addition, the operating frequency (logic switching rate) affects the power dissipation of some types of digital ICs.

Power dissipation is expressed in watts per gate and usually runs in the microwatt (10^{-6}) or milliwatt (10^{-3}) range. The total power dissipation in a digital circuit can be calculated by adding the dissipation values for all the gates used in the circuit.

Speed-Power Product

Speed and power are directly related in digital ICs. The faster the IC operates, the higher its power dissipation. Thus, in order to get high speed operation, you must sacrifice higher power dissipation. As you will soon discover, this is just one of the "trade-offs" that must be considered when designing digital circuits.

The speed-power relationship is expressed by a value called the **speed-power product**. This value is obtained by multiplying the propagation delay by the power dissipation. Note, however, that the propagation delay must be expressed in nanoseconds and the power dissipation in milliwatts. A low speed-power product is an advantage, since it means that the IC is relatively fast and dissipates minimal power.

Example 1-9

A digital IC specification gives the following information:

$t_{PLH} = 5\ ns$

Power dissipation = 10 mw

Calculate the speed-power product.

The speed-power product is obtained by multiplying the propagation delay in nanoseconds by the power dissipation in milliwatts. From the above specifications, the propagation delay is 5 nanoseconds and the power dissipation is 10 milliwatts. Thus, the speed-power product is:

$$5 \times 10 = 50$$

Speed

Whenever you consider the speed of anything, you must consider time. All operations in the real world take time; nothing happens instantaneously. The same is true of digital signals. Up to this point, you have been treating digital signals as if they made instantaneous changes. However, in reality, it takes time for digital signals to change state as well as time for a digital device to react to input signal changes.

RISE AND FALL TIME

A typical digital pulse is shown in Figure 1-23. Notice that the pulse is formed by a **leading edge** and a **trailing edge**. The time it takes the signal to change from a logic 0 to a logic 1 state is called **rise time**, or **t_r**. The time it takes the signal to change from a logic 1 to a logic 0 state is called **fall time**, or **t_f**. Notice that t_r and t_f are measured between the 10% and the 90% points of the leading and trailing edges of the signal.

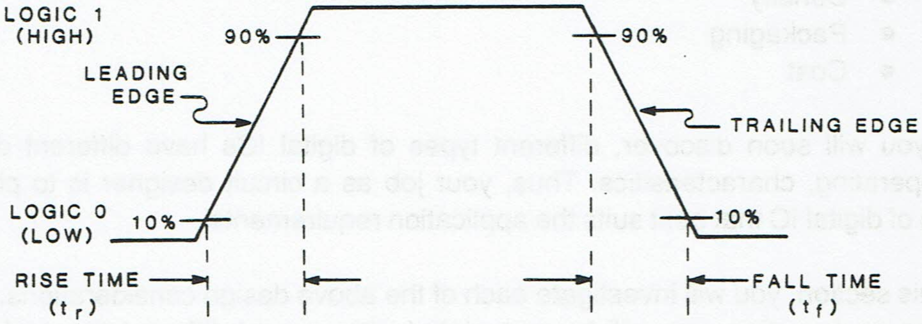


Figure 1-23
Rise and fall times of a digital signal.

PROPAGATION DELAY TIME

One of the most important characterisitcs of any digital circuit is its operational speed. For most applications, high speed operation is desirable. the speed characteristic of a digital circuit is called **propagation delay**, or **t_p**. Propagation delay is the amount of time it takes the output of a digital circuit to change after the inputs have changed. As an example, look at the inverter timing diagram shown

Answers

1. digital

2. See Figure 1-21

3A. eight

3B. sixteen

3C. thirty two

4. inverter

5. See Figure 1-22

6. NAND — This equivalency is called **DeMorgan's second theorem**.

7. False

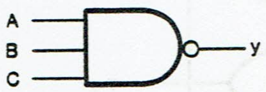
8. $y = \overline{A}B + A\overline{B}$.

9. XOR

10. XOR

11A. 0101

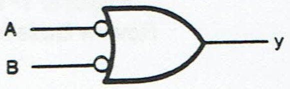
11B. 1010 — The input to the circuit is inverted when COMPLEMENT is high. When COMPLEMENT is low, the input is passed-on to the output. The COMPLEMENT line controls whether or not the input is inverted. For this reason, the circuit in Figure 1-20 is called a **controlled inverter**.



A	B	C	y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$y = \overline{ABC}$

Figure 1-21
Answer to Review Question 2.



A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

$y = \overline{A} + \overline{B}$

Figure 1-22
Answer to Review Question 5.

Self-Test Review

1. A two state signal is called a ______________ signal.

2. Develop a symbol, truth table, and Boolean expression for a 3-bit NAND gate.

3. How many input possibilities exist for the following:

   A. 3-bit gate? ____________
   B. 4-bit gate? ____________
   C. 5-bit gate? ____________

4. Tying the inputs of a NAND gate together produces a(n) ____________ gate.

5. Sketch the symbol and write the Boolean expression for a bubbled-OR gate.

Example 1-8

The input signals to an XOR gate are shown in the timing diagram of Figure 1-17. Sketch the corresponding output waveform.

You must begin by picking several timing points along the input waveforms. Of particular interest are those points when any of the inputs change their logic state. In Figure 1-17, I have picked ten points and labeled them t_0 through t_9. At each point, look at the corresponding input levels and, knowing the operation of an XOR gate, plot the output level. Thus, at point t_0 the output is low since both inputs are low. At point t_1 the output is high since odd parity exists on the input lines. At point t_2 the output is low since even parity is being applied to the input lines. Repeating this plotting process for each timing point, you get the output waveform shown in Figure 1-18.

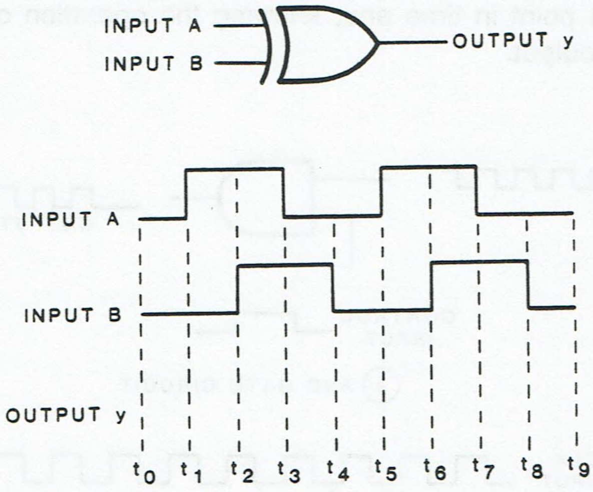


Figure 1-17

Sketching a timing diagram (Example 1-8).

Example 1-7

Although an XNOR gate recognizes even parity, it is commonly used in odd parity systems to generate the parity bit. Recall that in an odd parity system, the transmitted word must always contain an odd number of 1's. In Figure 1-15, the 7-bit XNOR gate monitors the code produced by the ASCII code generator. If the code is even parity, the XNOR gate generates a logic 1 in the most significant bit position to make the transmitted word odd parity. However, if the ASCII code generator produces an odd parity code, the XNOR gate generates a logic 0 such that the parity of the transmitted word remains odd.

What type of gate would you use to generate the parity bit in an even parity system?

An XOR gate.

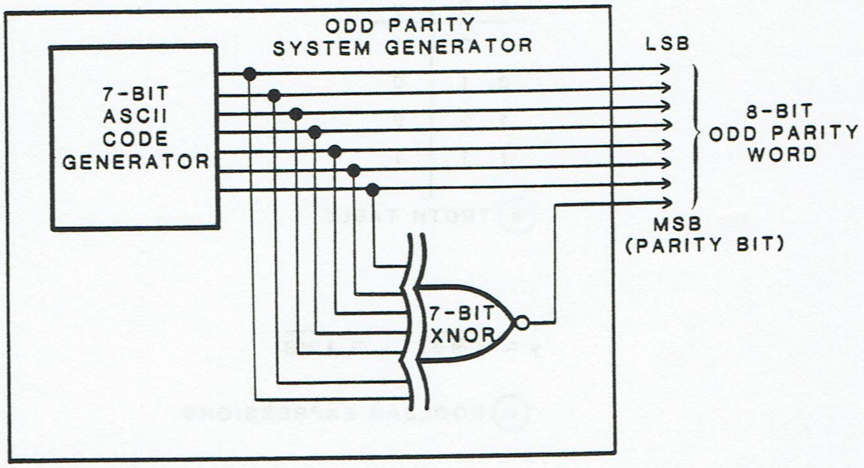


Figure 1-15
An XNOR gate is used to generate the parity bit in an odd parity system (Example 1-7).

Timing Diagrams

The operation of a digital logic circuit is often described using a graphical representation called a timing diagram.

TIMING DIAGRAM: A graph which shows the relationship between the input and output signals of a digital device or circuit.

One way to check for parity error is to use an XOR gate as shown in Figure 1-13. Here, an 8-bit XOR gate monitors the transmitted code in an even parity system. If a bit error occurs, the word changes to odd parity. Since the XOR gate recognizes odd parity, it generates a high output, illuminating an LED to indicate the error.

What must be done to the circuit in Figure 1-13 to detect errors in an odd parity system?

The simplest solution is to add an inverter to the output of the XOR gate. When this is done, the light emitting diode, or LED, is only illuminated when an even parity word is detected, thereby indicating an error condition.

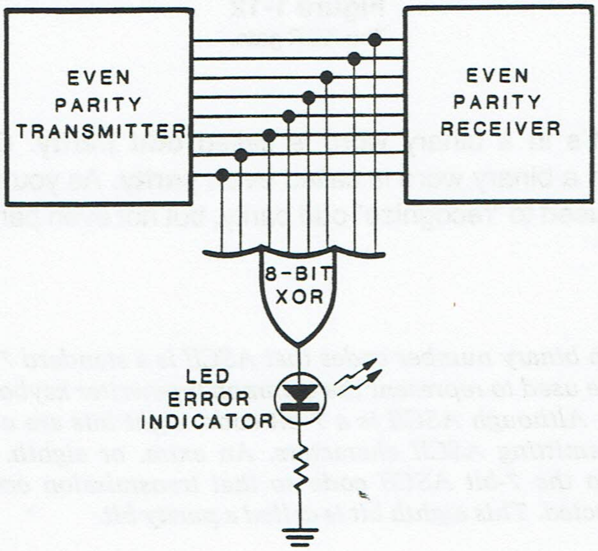


Figure 1-13
An XOR gate is used to check for errors in an even parity system (Example 1-6).

Example 1-5

*Show by means of truth tables that a bubbled AND gate and a NAND gate are **not** the same thing.*

In terms of Boolean algebra, you must show that $\bar{A}\bar{B}$ is **not** equal to $\overline{AB}$. From Example 1-4, you know that a bubbled AND gate is the same as a NOR gate. Thus, the truth table for bubbled AND gate expression $y = \bar{A}\bar{B}$ is:

A	B	y
0	0	1
0	1	0
1	0	0
1	1	0

Now, the truth table for NAND gate expression $y = \overline{AB}$ is:

A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

As you can see, it soon becomes obvious that the two truth tables are not the same. Therefore, $\bar{A}\bar{B} \neq \overline{AB}$.

The above example illustrates a very important idea: inverting a Boolean quantity does not necessarily mean that you can invert the individual variables within that quantity.

THE XOR GATE

The abbreviation XOR means **exclusive** OR. The XOR gate is different from the OR gate in that it "excludes" any even number of inputs from being high at the same time. The symbol, truth table, and Boolean expression for a 2- bit XOR gate are shown in Figure 1-12. Notice that the output is only high when an **odd** number of the input lines are high. In other words, if there are an odd number of 1's applied to the gate, the output is high. However, if there are an even number of 1's applied to the gate, the output is low. Notice from the truth table that an all zero input (00) also produces a low output, as would be expected.

Example 1-4

The symbol shown in the center of Figure 1-10 is called a bubbled AND gate, for obvious reasons. Show by means of truth tables that a ***bubbled*** *AND gate performs the same logic operation as a NOR gate.*

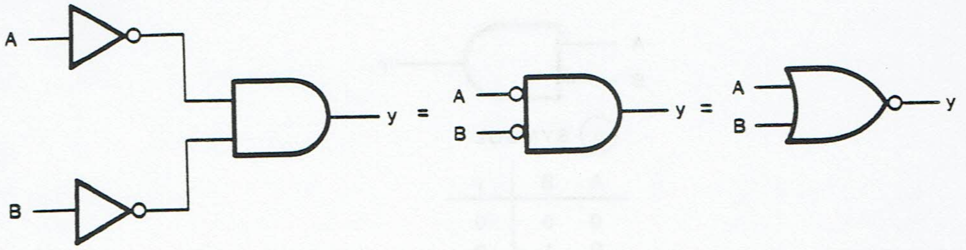


Figure 1-10

A bubbled AND gate performs the same operation as a NOR gate by DeMorgan's First Theorem.

Notice that a bubbled AND gate is simply an AND gate with its input lines inverted as shown on the left side of the figure. As a result, you can express its operation with Boolean algebra as $y = \overline{A}\overline{B}$. Notice that each input symbol has a separate "not" bar. Suppose you let y_1 represent the bubbled AND gate on the left side of the equation, and y_2 represent the NOR gate on the right side of the equation. Then,

$$y_1 = \overline{A}\overline{B}, \text{ and } y_2 = \overline{A+B}.$$

To determine the truth table for y_1, you must substitute all possible input combinations into the equation for y_1 as follows:

$$y_1 = \overline{A}\overline{B} = \overline{0}\overline{0} = 11 = 1$$

$$y_1 = \overline{A}\overline{B} = \overline{0}\overline{1} = 10 = 0$$

$$y_1 = \overline{A}\overline{B} = \overline{1}\overline{0} = 01 = 0$$

$$y_1 = \overline{A}\overline{B} = \overline{1}\overline{1} = 00 = 0$$

Example 1-3

Prove or disprove that $\overline{A+A} = \overline{A}$.

Notice that the left side of the equation is a NOR operation, and the right side of the equation is an inverter operation. Now, the easiest way to prove that two Boolean expressions are equal is to show that they have the same truth table. Suppose you let $y_1 = \overline{A+A}$, and $y_2 = \overline{A}$. To get the truth table for y_1 , you must perform a NOR operation on one input variable. The single input, A, can have two possible logic states, 0 and 1. Thus,

$$y_1 = \overline{A+A} = \overline{0+0} = \overline{0} = 1$$

$$y_1 = \overline{A+A} = \overline{1+1} = \overline{1} = 0$$

To determine the output state, you must substitute the input(s) and perform the required logic operations. Therefore, if $y_1 = \overline{A+A}$, the truth table for y_1 is:

A	y_1
0	1
1	0

Notice that this is the same as the truth table for an inverter. And, since y_2 is simply an inverter operation, the two truth tables are the same, thereby proving that $y_1 = y_2$, or $\overline{A+A} = \overline{A}$.

It is important to note what the above relationship says. In symbols, it says that the two symbols shown in Figure 1-8 are equivalent. In words, it says that you can connect, or tie, the input lines of a NOR gate together to get an inverter.

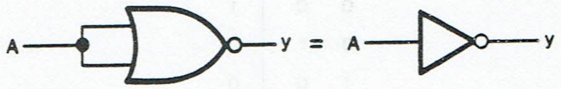


Figure 1-8
Connecting, or "tying," the inputs of a NOR gate together produces an inverter operation (Example 1-3).

The Boolean symbol for the OR operation is the plus, or +, symbol. The plus symbol is used since, if you think of the operation as addition, you will get the correct output. For instance, if A=1 and B=0, then A + B = 1 + 0 = 1. From the truth table you can see that this is the correct output. However, do not get the OR operation confused with addition, since they are two different operations. When you see a plus (+) sign in Boolean algebra, it means OR and not addition. To prevent possible confusion, some texts use the "V" symbol to represent an OR operation. This is convenient when a digital circuit is designed to perform both ORing and addition operations, as in computer circuits. However, in this text we will always use the "+" symbol to designate an ORing operation.

Example 1-2

Develop the symbol, truth table, and Boolean expression for a 3-bit OR gate.

As compared to a 2-bit OR gate, a 3-bit OR gate has three input lines rather than two. Thus, you simply add a third input line, C, to the symbol as shown in Figure 1-6.

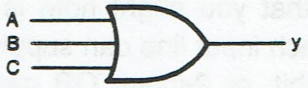


Figure 1-6
A 3-bit OR gate (Example 1-2).

Since there are three inputs to the gate, there are eight possible logic combinations that can be applied to the gate. These eight possibilities are shown on the input side of the truth table below:

A	B	C	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Notice that there are two possible symbols for the inverter. Each symbol consists of a triangle and a "bubble." The bubble comes either before or after the triangle. Regardless of where it is located, when you see a bubble, think "invert." In practice, the two inverter symbols are used interchangeably.

The truth table in Figure 1-3 describes the operation of the inverter. Since there is only one input, there are only two possible input values, 1 and 0. These possible input values are listed on the left side of the table. The corresponding output values are listed on the right side of the table. Notice the input is labeled **A** and the output is labeled **y**. In this course, upper case letters from the beginning of the alphabet, such as A, B, C, and D, will normally be used to designate input signals. Lower case letters at the end of the alphabet, such as x, y, and z, will be used to designate output signals.

Finally, the Boolean expression in Figure 1-3c provides a mathematical description of the gate's operation. The Boolean symbol for invert is the NOT, or "$\overline{\ \ }$", symbol. Thus, if you label the output y and the input A, then $y = \overline{A}$ is a Boolean expression that is used to describe the inverter operation. This expression is read: "y equals NOT A." For this reason, the inverter is sometimes called a NOT gate.

Boolean algebra will be discussed in more detail later, when you are required to simplify your circuit designs. For now, you will only need to concern yourself with those fundamental Boolean symbols and expressions that are used to describe the basic logic gates.

Example 1-1

The circuit in Figure 1-4 consists of two inverters connected in series. Determine the truth table and Boolean expression that describe the circuit.

A logic 1 input to the first inverter produces a logic 0 at its output. This output is input to the second inverter, which inverts it back to a logic 1. Consequently, the circuit does not produce any logic change from input to output. Thus, the corresponding truth table is:

A	y
1	1
0	0

Since a digital system is a two state system, the binary number system can be used to represent digital signals. Recall that the binary number system is a two state positional weight number system made up of 1's and 0's. The two levels of a digital signal can be equated to a 1 and 0, thereby allowing any number to be represented using the binary system. In most cases, the highest voltage level is used to represent a binary 1 and the lowest voltage level is used to represent a binary 0. For this reason, a binary 1 in a digital circuit is referred to a logic 1, or high, level. Likewise, a binary 0 is called a logic 0, or low, level.

It is easy to manufacture digital circuits since many electronic devices exhibit two state operation. A diode is either conducting or not conducting, a capacitor is either charged or not charged, and a transistor can be either saturated or cut-off. In addition, digital circuits are not as noise prone as analog circuits, since sufficient voltage levels are used to prevent most noise from affecting the digital value. Finally, digital circuits are much faster and require less accurate and smaller power supplies than their analog counterparts.

I should mention, however, that the digital world is more or less an artificial world. For digital circuits to communicate with the real analog world, conversions must be made between the two types of signals. For now, you will concentrate on the world of digital. Let's begin with the foundation of a digital circuit, the logic gate.

Digital Logic Gates

A logic gate operates on one or more digital input signals and produces one output signal. These gates are made up of resistors, diodes, transistors, and capacitors that act on the input signals to produce the required output. In this course, you will not concern yourself with the internal design of digital logic gates and devices. Rather, you will treat them as functional "black boxes," emphasizing their input/output characteristics in order to design working digital circuits.

The important idea in the above definition is that an analog signal can take on any value between some minimum and maximum level. The voltage waveforms shown in Figure 1-1 are all examples of analog signals. If a computer were designed around analog signals, numbers would have to be represented with different voltage or current levels. For example, the number 1 might be represented by 1 volt, 2 by 2 volts, and so on. The obvious disadvantage of this scheme is that it would require very large voltage levels to represent large numbers.

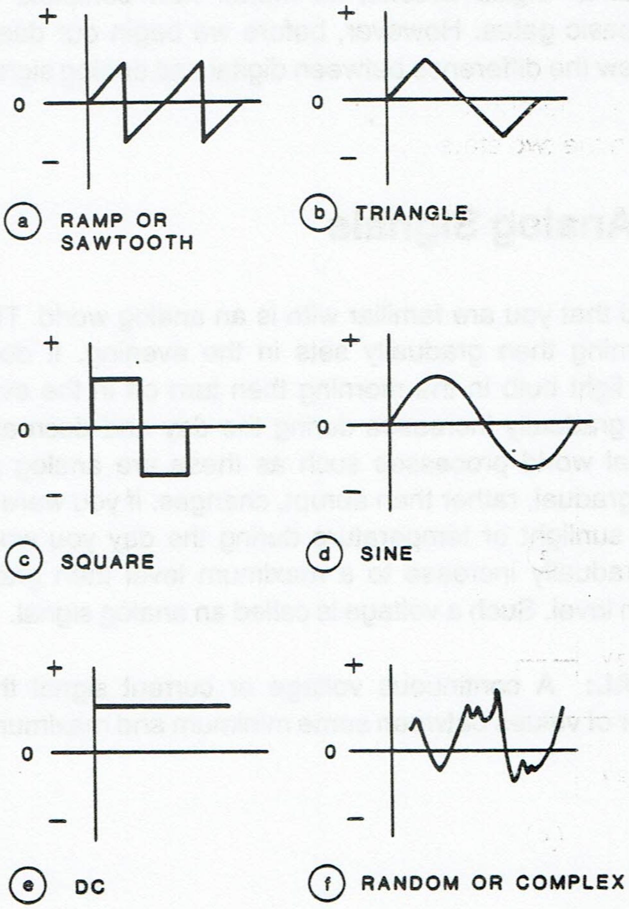


Figure 1-1
Types of analog signals.

UNIT ACTIVITY GUIDE

☐ Read section on “Basic Logic Gates.”

☐ Answer Self-Test Review Questions 1-11.

☐ Read section on “Digital Design Considerations.”

☐ Answer Self-Test Review Questions 12-19.

☐ Read section on “Digital IC Technology.”

☐ Answer Self-Test Review Questions 20-26.

☐ Study Summary.

☐ Perform Experiments 1 through 4 in Unit 8.

☐ Complete Unit Examination.

☐ Check Examination Answers.

UNIT OBJECTIVES

When you have completed this unit, you should be able to:

1. Develop a symbol, truth table, and Boolean expression for each of the seven basic logic gates.
2. Prove or disprove Boolean relationships.
3. Construct simple logic gate circuits.
4. Trace the logic through simple gate circuits in order to develop circuit truth tables.
5. Sketch timing diagrams for simple logic gate circuits.
6. Explain the difference between even and odd parity.
7. Design digital circuits to generate and recognize even and odd parity.
8. Explain the following design considerations as related to a digital circuit:
    - A. Speed
    - B. Power Dissipation
    - C. Fan Out
    - D. Noise Immunity
    - E. Density
    - F. Packaging
    - G. Cost

CONTENTS

Introduction . . . . . Page 1-3
Unit Objectives . . . . . Page 1-4
Unit Activity Guide . . . . . Page 1-6
Basic Logic Gates . . . . . Page 1-7
Digital Design Considerations . . . . . Page 1-31
Digital IC Technology . . . . . Page 1-43
Summary . . . . . Page 1-64
Unit Examination . . . . . Page 1-65
Examination Answers . . . . . Page 1-71

UNIT 2

COMBINATIONAL LOGIC DESIGN USING SSI DEVICES

CONTENTS

Introduction . . . . Page 2-3
Unit Objectives . . . . Page 2-5
Unit Activity Guide . . . . Page 2-6
The Design Process . . . . Page 2-7
Problem Definition and Representation . . . . Page 2-18
Simplification . . . . Page 2-28
SSI Implementation . . . . Page 2-57
Summary . . . . Page 2-74
Unit Examination . . . . Page 2-76
Examination Answers . . . . Page 2-81

INTRODUCTION

You are now ready to begin designing working digital logic circuits. The two basic logic circuits that you will design in this course are **combinational logic** and **sequential logic** circuits.

Combinational logic circuits make logic decisions. They are called combinational circuits, since they represent various combinations of the basic logic gates. If you think about it, a gate is a simple decision making element. A given gate looks at its binary inputs and generates a binary output that represents the decision it has made, according to its operation. Combining two or more gates in a combinational logic circuit allows more sophisticated decision making. In terms of a "black-box", a combinational logic circuit accepts multiple binary inputs and generates one or more output decisions as shown in Figure 2-1a.

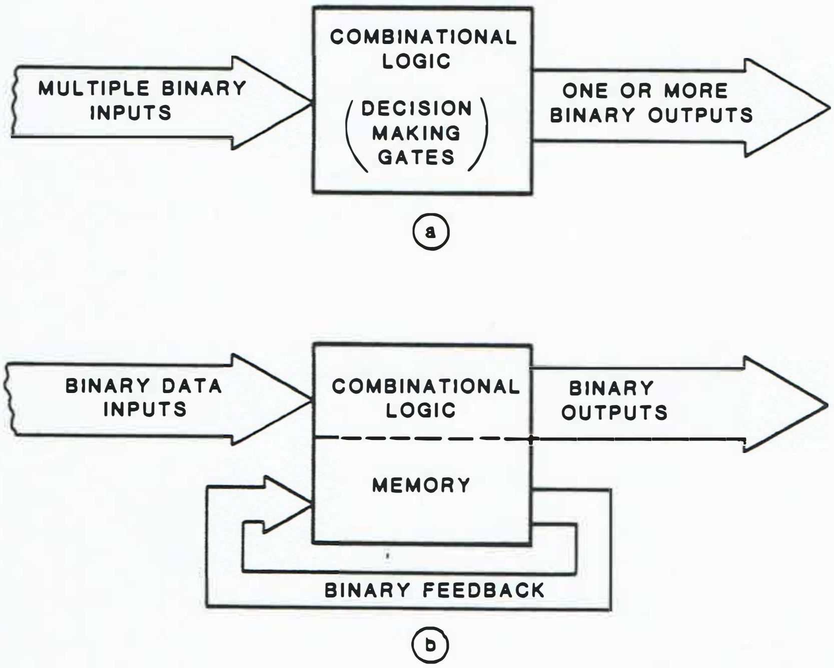


Figure 2-1
A combinational logic circuit **a**, and a sequential logic circuit **b**.

In this unit, you will learn how to design such circuits using SSI devices. Then, in Unit 3, you will learn how to design more complex combinational logic circuits using MSI and LSI devices.

A sequential circuit can also make decisions, but it has one very important feature not found in a combinational logic circuit — **memory**. Thus, a sequential logic circuit includes combinational logic as well as memory. The black-box in Figure 2-1b shows the functional characteristics of a sequential logic circuit. Notice that its inputs consist of external binary data, and feedback signals developed within the circuit itself. The outputs of a sequential circuit are a function of the binary inputs, the feedback signals, the data stored in its internal memory elements, and the design characteristics of the circuit. You will learn how to design sequential logic circuits in Unit 4. I will defer further discussion of the design and use of these circuits until that time. At this time, let's look at a general design process that can be applied to the design of any digital circuit, combinational or sequential.

UNIT OBJECTIVES

When you have completed this unit, you should be able to:

1. Describe the two basic types of logic circuits.
2. Explain the black-box design concept.
3. Develop a truth table for both a common cathode and common anode seven-segment LED.
4. Explain the seven basic steps of the SSI design algorithm.
5. Define inputs, outputs, and I/O logic for simple digital applications problems.
6. Develop truth tables for digital application problems.
7. Write Boolean expressions from truth tables.
8. Simplify Boolean expressions using both Boolean algebra and Karnaugh mapping.
9. Implement Boolean expressions using standard AND/OR gates.
10. Convert AND/OR logic designs to NAND/NOR/XOR designs.
11. Implement logic designs using standard TTL devices.

UNIT ACTIVITY GUIDE

☐ Read section on "The Design Process."

☐ Answer Self-Test Review Questions 1-9.

☐ Read section on "Problem Definition and Representation."

☐ Answer Self-Test Review Questions 10-17.

☐ Read section on "Simplification."

☐ Answer Self-Test Review Questions 18-25.

☐ Perform Experiment 5 in Unit 8.

☐ Read section on "SSI Implementation."

☐ Answer Self-Test Review Questions 26-31.

☐ Study Summary.

☐ Perform Experiments 6 and 7 in Unit 8.

☐ Complete Unit Examination.

☐ Check Examination Answers.

THE DESIGN PROCESS

Commercial digital designs don't just happen, they are well planned and thought-out. Even simple designs must be planned and organized from beginning to end. If not, the results are often disastrous. Whether the design involves one engineer or several hundred engineers, the design process must follow a well organized development plan. In fact, the planning phase is probably the most important part of any design. Unfortunately, it is the part that is most often overlooked. Good designs are always the result of good planning.

Product Development

Just about all commercial products follow the development path shown in Figure 2-2. In this and the next unit, you will learn how to design various digital circuits and prepare the circuit design specifications. However, as you can see from Figure 2-2, the design process is by no means complete at this stage. The subsequent stages of prototyping, debugging, manufacturing, testing, and customer evaluation must be performed in order to complete the design process. These later stages of the design process will be discussed in subsequent units.

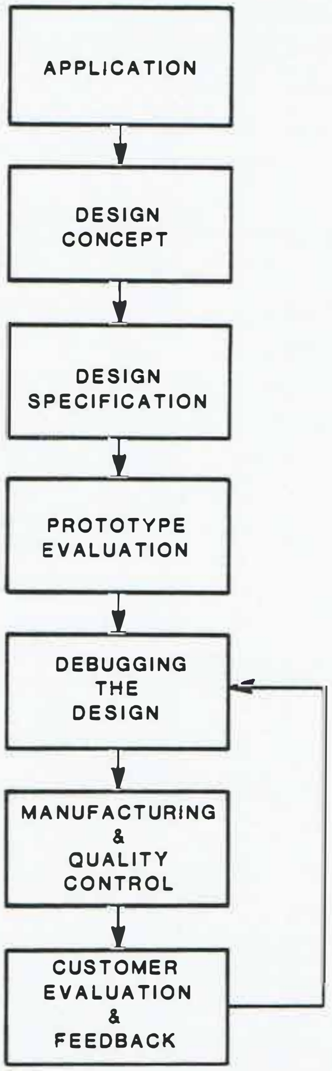


Figure 2-2

The design process and product development path.

As you can see, product development begins with an application which must be translated into a design concept. The design concept requires the designer to study the application and determine what the application dictates for the design. One of the most useful tools available to the designer at this stage is the **black-box** concept.

The Black-Box Concept

The idea behind the black-box concept is to divide a complex design problem down into manageable subproblems. Many times, the entire system to be designed is too complex for your mind to grasp all at one time. By breaking the system down into manageable subsystems, you can ignore details that are unimportant at a given level and concentrate your efforts on those details that are important.

Using the black-box concept, each subsystem level is treated as a box where the only concern is the function of the box and its input/output characteristics. This idea is illustrated by Figure 2-3. In digital designs, the black-box concept is applied from the overall system box down to the black-boxes that are used to represent the individual gates within the logic circuit. Thus, the digital design concept progresses from the system box down to the black-boxes containing the circuit details. Once the design concept has been developed using black-boxes, the actual design then proceeds in the opposite direction, from circuit details to total system integration.

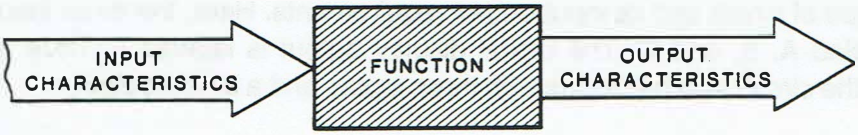


Figure 2-3

The black-box concept breaks a system down into black boxes where function and I/O characteristics for a given box are the only concerns.

Let's look at how the black-box concept might be applied to a simple digital design. Suppose the application is to design a digital voting machine. For illustrative purposes, we will keep our voting machine relatively simple. Thus, suppose our machine is to indicate if a particular issue passes or fails from the yes/no votes of just three voters.

The application task immediately suggests the overall system black-box shown in Figure 2-4. Notice that the black-box shows the function of the system as well as any system input/output requirements. At this level, you are not concerned with the circuit details. The major concern is the overall function of the system.

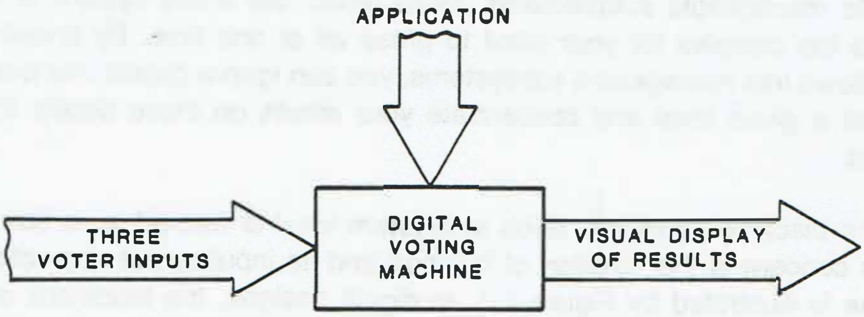


Figure 2-4

The system black-box for a digital voting machine.

From a single system black-box, the design is broken down into additional black-boxes that describe the functional requirements of the various subsystems as shown in Figure 2-5. At this level, the black-boxes provide more detail relating to the type of circuit and its input/output requirements. Here, the three input voters are labeled A, B, and C. The digital display output is labeled y. Thus, you can think of the circuit as a black-box with three inputs and a single output.

More detail is provided at the third level, shown in Figure 2-6. Notice that the actual circuit details are described at this final level. Of course, a more complex application would require more levels of black-boxes.

Thus, the development process for our simple voter circuit is complete through the design concept stage using black-boxes. Going back to Figure 2-2, you see that the next step is to prepare the actual design specifications. In practice, a set of design specifications are prepared for each black-box within the system, increasing in detail until the final specifications represent the actual logic circuit diagram, input/output devices, and any required interface connections. So, the next stage in the design process is to prepare a detailed circuit design.

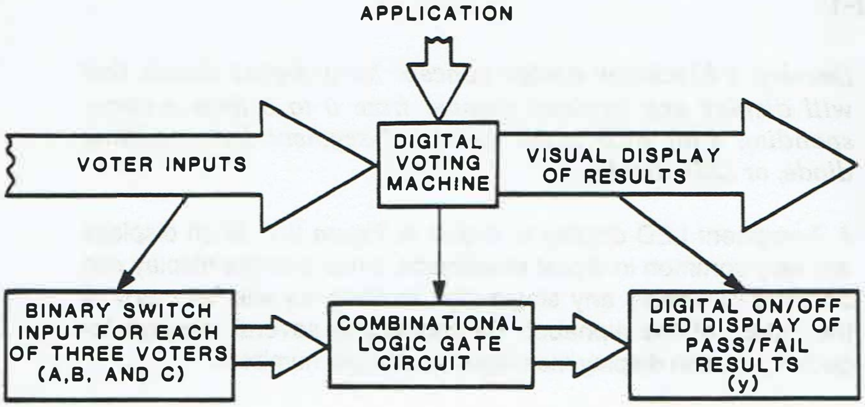

APPLICATION
VOTER INPUTS
DIGITAL VOTING MACHINE
VISUAL DISPLAY OF RESULTS
BINARY SWITCH INPUT FOR EACH OF THREE VOTERS (A,B, AND C)
COMBINATIONAL LOGIC GATE CIRCUIT
DIGITAL ON/OFF LED DISPLAY OF PASS/FAIL RESULTS (y)

Figure 2-5

The next level of black-boxes describes the functional requirements of the various subsystems.

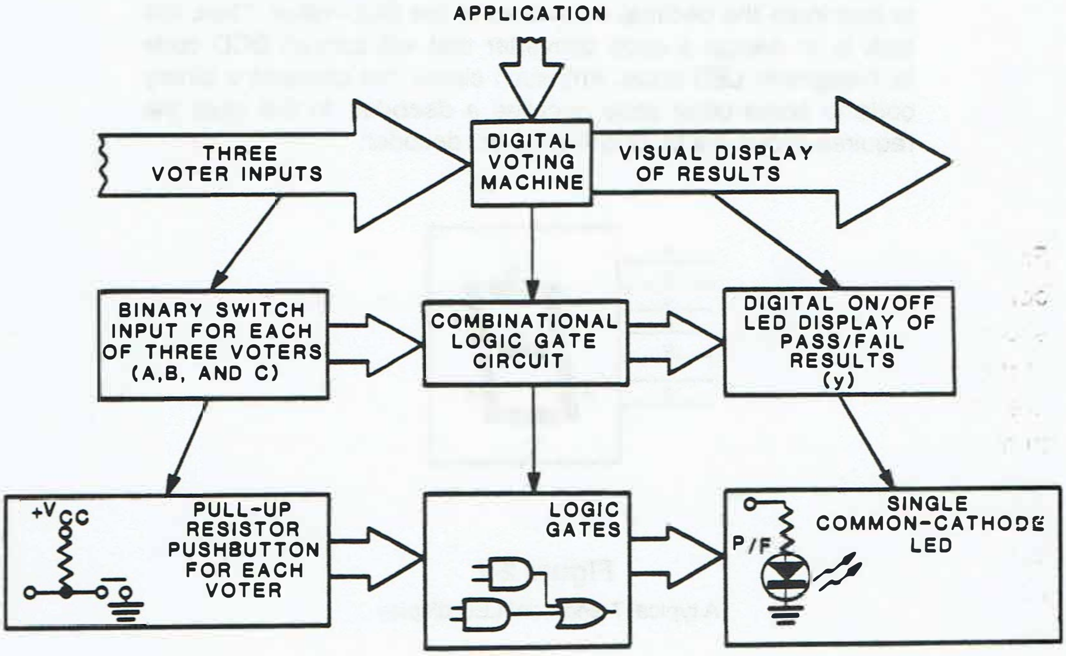

APPLICATION
THREE VOTER INPUTS
DIGITAL VOTING MACHINE
VISUAL DISPLAY OF RESULTS
BINARY SWITCH INPUT FOR EACH OF THREE VOTERS (A,B, AND C)
COMBINATIONAL LOGIC GATE CIRCUIT
DIGITAL ON/OFF LED DISPLAY OF PASS/FAIL RESULTS (y)
+Vcc
PULL-UP RESISTOR PUSHBUTTON FOR EACH VOTER
LOGIC GATES
P/F
SINGLE COMMON-CATHODE LED

Figure 2-6

The final level of black-boxes provides the circuit details.

Example 2-1

Develop a black-box design concept for a digital circuit that will display any decimal number from 0 to 9 from a corresponding 4-bit BCD input using a 7-segment light emitting diode, or LED, display.

A 7-segment LED display is shown in Figure 2-7. Such displays are very common in digital electronics, since a single display can be used to display any single-digit number, as well as many of the letters of the alphabet. By connecting several displays together, you can display messages and larger numbers.

Observe from Figure 2-7 that the display is made-up of seven individual LED segments, labeled "a" through "g". There are seven input lines to the display, one for each individual LED segment. To display a number from 0 to 9, the correct segment pattern must be illuminated. To display the number 3, for example, segments a, b, c, d, and g must be illuminated.

The application task requires that a digital circuit be designed to convert a 4-bit BCD code into a code that will cause the display to illuminate the decimal equivalent of the BCD value. Thus, the task is to design a code converter that will convert BCD code to 7-segment LED code. Any such circuit that converts a binary code to some other code is called a **decoder**. In this case the required circuit is a BCD to 7-segment decoder.

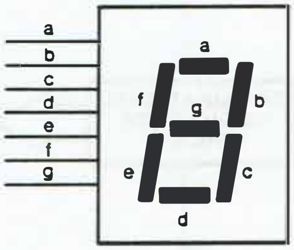


Figure 2-7

A typical 7-segment LED display.

The required black-box is shown in Figure 2-8. The inputs to the circuit have been labeled A, B, C, and D for each bit position in a 4-bit BCD value. There are seven outputs from the circuit, labeled "a" through "g". Each output must be connected to the respective segment input line on the LED.

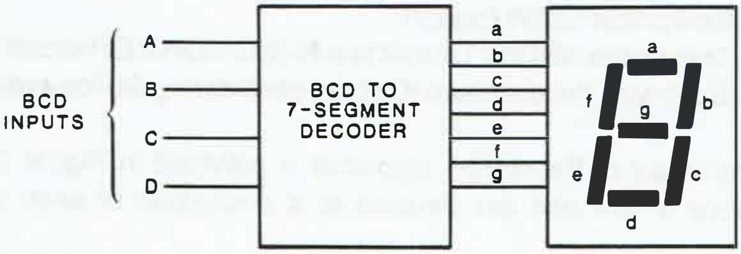


Figure 2-8

The black-box concept for a BCD to 7-segment decoder.

The SSI Design Algorithm

The remainder of this unit is devoted to teaching you the techniques required to successfully design combinational logic circuits using SSI devices. The simple voter circuit design concept developed in Figure 2-6 has led you directly into the circuit design and specification stage. The task now is to develop the actual circuit design. To accomplish this, all you need to do is follow a step-by-step design procedure that I call the "SSI design algorithm". The SSI design algorithm will always generate the simplest combinational logic circuit that will satisfy the given application. The algorithm steps are as follows:

Step 1: **I/O Definition**
Define all inputs and outputs from the application task and the design concept.

Step 2: **Logic Definition**
Define the meaning of a logic 0 and a logic 1 for each input and output specified in step 1.

Step 3: **Truth Table**
Develop a truth table from the design concept and your definitions in steps 1 and 2.

Step 4: **Boolean Expression**
Develop a Boolean expression from the truth table in step 3.

Step 5: **Simplification**
Simplify the Boolean expression derived in step 4.

Step 6: **AND/OR Design**
Construct an AND/OR circuit from the simplified Boolean expression.

Step 7: **NAND/NOR/XOR Design**
Convert the AND/OR circuit to a NAND/NOR/XOR circuit as required to produce the minimum IC count considering device availability.

A flowchart summary of the design algorithm is provided in Figure 2-9. The remaining sections in this unit are devoted to a discussion of each design step in this figure.

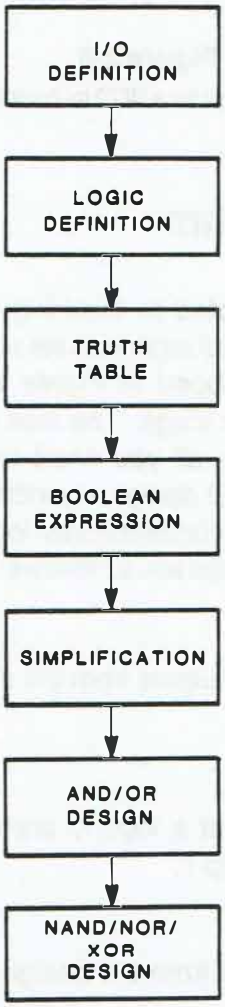


Figure 2-9
A flowchart summary of the SSI design algorithm.

Self-Test Review

1. The two basic types of logic circuits are ____________________ and ____________________.

2. Digital networks that make decisions are called ________ logic circuits.

3. At which stage of the design process do you use the black-box concept?

____

4. What design information is provided by a system level black-box?

____

____

____

5. Which input lines in Figure 2-7 must be activated to display the letter "H"?

____

6. Suppose a logic 1 is needed to illuminate a segment in Figure 2-7. Develop a truth table for the "d" segment to display the decimal values 0 through 9.

7. Assuming a logic 0 causes a segment to illuminate in Figure 2-7, develop a truth table for the "d" segment to display the decimal values 0 through 9.

8. What type of logic gate is required to convert the truth table in question 6 to the truth table in question 7? ____________.

9. The SSI design algorithm is used to complete the ____________ stage of the design process.

Answers

1. combinational (decision making) and sequential (memory)

2. combinational

3. When developing the design concept during the digital design process.

4. Function of the circuit. Input characteristics. Output characteristics.

5. b, c, e, f, and g

6.

VALUE	d
0	1
1	0
2	1
3	1
4	0
5	1
6	1
7	0
8	1
9	0

7.

VALUE	d
0	0
1	1
2	0
3	0
4	1
5	0
6	0
7	1
8	0
9	1

8. Inverter

9. specification

PROBLEM DEFINITION AND REPRESENTATION

In this section, you will learn how to apply the first four steps of the SSI design algorithm: I/O definition, logic definition, truth table, and Boolean expression. Although the first two steps, I/O and logic definition, might seem simple and rather trivial, they are probably the two most important steps in the whole process. Do not take them lightly! I have found that many times students tend to overlook these two simple steps, only to find out later that their completed design does not work and wonder, why?

Since the intent of this course is to teach you how to design working digital circuits, the best way to learn is by following through a few simple design examples. So, let's begin with the voter circuit presented in the last section.

I/O Definition

From the application task and the design concept, you should have a good idea of what the circuit inputs and outputs must be. For example, consider the simple voter circuit discussed earlier. The application task and design concept have dictated that there must be three inputs, one for each voter, and a digital display output to indicate whether an issue passes or fails. The three input voters have already been designated as A, B, and C, while the digital output has been labeled y. So, let's provide three binary switches on the circuit input, one for each voter to indicate a "yes" or "no" vote. A single light emitting diode, or LED, on the circuit output will indicate whether an issue passes or fails.

A given voter can activate his/her binary switch one way or the other for a yes or no vote. The single LED must then indicate if the issue passes or fails by illuminating or not illuminating. Suppose the design specifications dictate that the LED must illuminate when an issue passes.

In summary, you have:

Inputs: 3 binary switches, one for each voter.

Outputs: 1 single LED that must illuminate if an issue passes.

Logic Definition

Now that you have defined the circuit inputs and outputs, you must define the meaning of a logic 0 and a logic 1 for each input and output. It is important that the logic for **each** input and output be defined, since you will use this in the next step of the process.

Returning to the voter circuit example, suppose you let a logic 1 from a binary switch represent a "yes" vote and a logic 0 represent a "no" vote. This logic definition will be applied to each of the binary input switches.

Next, suppose you let a logic 1 on the circuit output indicate that an issue passes. Conversely, a logic zero will indicate that the issue fails.

In summary, your logic definitions for the voter circuit are as follows:

Binary Switch Inputs:	Logic 1 "yes" vote. Logic 0 "no" vote.
LED Output:	Logic 0 issue fails. Logic 1 issue passes.

I should point out that it doesn't matter how the logic is defined. In other words, you could have just as easily let a logic 0 be a "yes" vote and a logic 1 be a "no" vote. Likewise, a logic 0 could be used to indicate that an issue passed and a logic 1 that an issue failed. The choice is completely yours and will not affect the circuit operation, as long as you stick with your definitions throughout the design process.

The way we have defined the logic is referred to as **positive logic**. It just seems natural to let a 0 mean nothing, such as a "no" vote, and a 1 indicate something, like a "yes" vote. This is the logical way that you and I think. On the other hand, **negative logic** would dictate that a 0 be a "yes" vote and a 1 be a "no" vote. However, this doesn't seem natural since we are not taught to think in terms of negative logic, even though it will not affect the circuit operation.

There are many cases in digital electronics where negative logic is employed. When this is the case, a logic 0 is said to be the **active state**. Negative logic is often used since it requires less power to generate a logic 0, or ground, than it does to generate a +5 volt level. Stated in electrical terms, it is easier to **sink** current to ground than it is to **source** current. So, remember that a logic 0 means more than just nothing. It is half the binary number system and can be used the same way as a logic 1 when defining active logic states.

The Truth Table

The truth table step requires that you apply your I/O and logic definition from steps 1 and 2 directly to the application task. Recall that a truth table simply shows, in logic form, what the circuit will do. In other words, it shows what the circuit output(s) will be for all the input possibilities. So, to construct a truth table, the first step is to set-up a blank table which lists all the input possibilities. In the voter circuit example, there are three binary inputs: A, B, and C. Thus, there are 2^3, or 8, input possibilities. They are:

A	B	C	y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Now, to fill in the table, you must look at how you defined your logic in step 2. The truth table must be consistent with your logic definitions. For example, a 0 0 0 input would mean three "no" votes, and thus require a 0 output to indicate that the issue failed. Consequently, you must enter a 0 in the table for y, opposite the 0 0 0 input.

The next possibility is a 0 0 1 input, or "no", "no", "yes" vote. Again, the issue fails and y must be 0.

The only time that the output, y, is a logic 1 is when an issue passes. This only happens when there are two or more "yes" votes, or logic 1's, on the circuit inputs. Applying this idea to the truth table, you get:

A	B	C	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The Boolean Expression

After you construct the truth table, the next step in the design process is to write a Boolean expression that describes the circuit operation. The Boolean expression is derived directly from the truth table. The idea is simple: each place you see a logic 1 output, look at the corresponding binary input and write down the **fundamental product** representation of the input.

The fundamental product representation of a binary value is obtained by complementing an input variable if it is a logic 0. If it is a logic 1, the variable is left uncomplemented. For instance, if input A is 0, it is written as $\overline{A}$; however, if A is 1 it is written as A. Thus, the input combination A B C = 0 0 0 would be written in fundamental product form as $\overline{A}\,\overline{B}\,\overline{C}$, while the input combination A B C = 0 0 1 must be written as $\overline{A}\,\overline{B}C$. Table 2-1 summarizes the fundamental product representations for each possibility of three input variables.

Table 2-1: Fundamental Product Representations for Three Input Variables

Input A B C	Fundamental Product
0 0 0	$\overline{A}\,\overline{B}\,\overline{C}$
0 0 1	$\overline{A}\,\overline{B}C$
0 1 0	$\overline{A}B\overline{C}$
0 1 1	$\overline{A}BC$
1 0 0	$A\overline{B}\,\overline{C}$
1 0 1	$A\overline{B}C$
1 1 0	$AB\overline{C}$
1 1 1	ABC

Now, back to the problem at hand. From the truth table we developed for the voter circuit, you can see that there are four logic 1 outputs. Writing the corresponding inputs in fundamental product form, you get $\overline{A}BC$, $A\overline{B}C$, $AB\overline{C}$, and ABC. To obtain a Boolean expression from a truth table you simply OR the fundamental products. Therefore, the Boolean expression for the voter circuit is:

$$y = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

This method is referred as the **sum-of-product**, or SOP, method for obtaining a Boolean expression from a truth table. The name "sum-of-product" comes from the idea that, if you look at the equation arithmetically, the individual terms are products which are summed to form the equation. Of course, you know that this terminology is not technically correct, since the equation is a logic equation, not an arithmetic equation.

Example 2-2

Perform the first four design steps required to design a BCD to 7-segment decoder circuit that will illuminate the "e" segment of a ***common anode*** *7-segment display.*

Step 1: Define Inputs and Outputs

From the black-box design concept developed in Example 2-1, the inputs are A,B,C, and D, which make up the bits of a 4-bit BCD input value. Let's use four binary switches to generate these BCD values.

The circuit outputs are the seven lines, "a" through "g", required to illuminate the LED segments. However, in this example you are only interested in the "e" segment. Thus, the only output required for this circuit is output line "e".

In summary, the inputs and outputs have been defined as follows:

Inputs: Four binary switches (ABCD), one for each bit in a 4-bit BCD value.

Outputs: The "e" segment line of a 7-segment LED.

Step 2: Logic Definition

The input logic is rather straight forward. The BCD input value, ABCD, must range from 0000 to 1001. Let's assume that input A generates the most significant bit, or MSB, and input D generates the least significant bit, or LSB.

As far as the output logic is concerned, a given segment of a 7-segment display is illuminated by placing a logic 0 or a logic 1 on its respective segment line. If the segments require a logic 0 to illuminate, the display is called a **common anode** display. Conversely, if the segments require a logic 1 to illuminate, it is called a **common cathode** display. In this example, you are using a common anode display which requires logic 0's to illuminate the segments. As a result, your circuit must generate a logic 0 each time the "e" segment is to illuminate.

In summary, the required logic definition is:

Inputs:	Four BCD input bits, ABCD, ranging from 0000 to 1001.
Outputs:	A logic 0 to illuminate the "e" segment of a common anode 7-segment display when required as the result of a given BCD input value.

Step 3: Truth Table

The required truth table is developed from the problem statement and logic definition. The input values are listed as you have defined them in step 2. Then, you determine the output logic according to the application task. The resulting truth table is:

Decimal Value	BCD A B C D	Segment "e"
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	0
3	0 0 1 1	1
4	0 1 0 0	1
5	0 1 0 1	1
6	0 1 1 0	0
7	0 1 1 1	1
8	1 0 0 0	0
9	1 0 0 1	1

Notice that the truth table shows a logic 0 output each time the "e" segment must be illuminated.

Step 4: Boolean Expression

The Boolean expression is obtained by ORing the fundamental products of the above truth table. The fundamental products are: $\overline{A}\,\overline{B}\,\overline{C}D$, $\overline{A}\,\overline{B}CD$, $\overline{A}B\overline{C}\,\overline{D}$, $\overline{A}B\overline{C}D$, $\overline{A}BCD$, and $A\overline{B}\,\overline{C}D$. Thus the Boolean sum-of-products equation is:

$$e = \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + A\overline{B}\,\overline{C}D$$

You should observe that, even though a logic 0 output is the active state, the Boolean expression is still written using the logic 1 outputs from the truth table. This is because fundamental product terms are only defined for a logic 1 output.

NUMERICAL REPRESENTATION OF A BOOLEAN EXPRESSION

Many times it is more convenient to write a Boolean expression in **numerical form**. The numerical representation of an SOP equation is written as:

$$f(A,B,C,D) = \Sigma(n_0,n_1,n_2,\ldots)$$

Where: f(A,B,C,D) means "a function of the input variables A, B, C, and D."

Σ (sigma) is a symbol that means "the summation of."

n_0, n_1, n_2, etc. are the decimal equivalents of the fundamental product terms of the variables A, B, C, and D.

As an example, look at the truth table for the "e" segment in Example 2-2. The decimal equivalents of the fundamental product terms are 1, 3, 4, 5, 7, and 9. Using the numerical representation, the associated Boolean expression is written as:

$$f(A,B,C,D) = \Sigma\,(1,3,4,5,7,9)$$

Notice that the numerical representation "says" the same thing as a standard SOP expression. The numerical representation says that "the e segment is a function of the input variables A, B, C, and D, which equals the summation of the fundamental product terms 1, 3, 4, 5, 7, and 9.

Self-Test Review

10. Explain why the first two steps in the SSI design algorithm, I/O definition and logic definition, are so important to the design process.

____

____

____

11. Using a logic 0 for an active state is called ________ logic.

12. Develop a truth table for the "f" segment of a common cathode LED to display the decimal values 0 through 9. Assume a BCD input of 0000 through 1001.

13. Write the SOP equation for the truth table you developed in question 12.

14. How many and what size AND and OR gates must be used to implement the Boolean expression developed for the voter circuit in this section.

15. Develop a truth table and SOP equation for the "f" segment of a common anode LED to display the decimal values 0 through 9. Assume a BCD input of 0000 through 1001.

16. Write the numerical representation for the SOP equations you derived in question 13 on line A, and question 15 on line B.

A. ______________________________

B. ______________________________

17. The numerical representation for a Boolean equation is:

$$f(w,x,y,z,) = \Sigma(0,9,13,15)$$

Write the equation in fundamental product form.

____

Answers

10. I/O and logic definition are extremely important to the design process since the logic truth table, equation(s), and circuit are derived directly from these two steps.

11. negative

12.

Decimal Value	BCD A B C D	Segment "f"
0	0 0 0 0	1
1	0 0 0 1	0
2	0 0 1 0	0
3	0 0 1 1	0
4	0 1 0 0	1
5	0 1 0 1	1
6	0 1 1 0	1
7	0 1 1 1	0
8	1 0 0 0	1
9	1 0 0 1	1

13. $f = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D} +$
$\overline{A}B\overline{C}D + \overline{A}BC\overline{D} +$
$A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}D$

14. Four 3-bit AND gates, one for each fundamental product term, driving a single 4-bit OR gate.

15.

Decimal Value	BCD A B C D	Segment "f"
0	0000	0
1	0001	1
2	0010	1
3	0011	1
4	0100	0
5	0101	0
6	0110	0
7	0111	1
8	1000	0
9	1001	0

$f = \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}C\overline{D} +$
$\overline{A}\,\overline{B}CD + \overline{A}BCD$

16A. $f = \Sigma(0, 4, 5, 6, 8, 9)$

16B. $f = \Sigma(1, 2, 3, 7)$

17. $f(w,x,y,z) = \overline{w}\,\overline{x}\,\overline{y}\,\overline{z} + w\overline{x}\,\overline{y}z +$
$wx\overline{y}z + wxyz.$

SIMPLIFICATION

After completing the first four design steps in the last section, you ended-up with a Boolean expression that describes the application circuit. The Boolean expression is a mathematical model for the circuit since it was derived from the truth table and, therefore, describes the logic operation of the circuit. As you are now aware from Unit 1, an AND/OR circuit can be developed directly from a Boolean expression. Thus, you might think that a working circuit can be constructed directly from the Boolean expression obtained in step 4. This is true, but the Boolean expression obtained from the truth table does not always produce the simplest and, therefore, the most economical circuit.

Do you recall what you spent many hours doing in algebra class? Once an algebraic equation was developed, you probably were instructed to reduce the equation to its simplest form. You did this by applying the various laws of algebra to eliminate unnecessary operations and terms. Likewise, since a Boolean expression is an algebraic expression, it can be reduced by applying the various laws of Boolean algebra. Unlike conventional algebra, the beauty of Boolean algebra is that you always see a direct relationship between the mathematics and the actual circuit.

What do you think happens to the resulting circuit as the Boolean expression is simplified? You guessed it, the circuit itself becomes simpler and less costly to produce. A simpler design results in fewer ICs, smaller printed circuit boards, and greater reliability. The costs you save with a simpler circuit are realized in both materials and time.

In the first part of this section, you will become acquainted with several of the common laws of Boolean algebra. These laws are different from the laws of conventional algebra, but can be applied in the same way to simplify Boolean expressions — the result being a simplified logic circuit.

You will soon discover that attempting to apply the laws of Boolean algebra to more complex expressions can get tricky and messy. Recall from conventional algebra that in many cases you had to somehow "see" which Law must be applied at a given point in order to end-up with the simplest equation. The same is true of Boolean algebra. Fortunately, there is a very powerful technique available for simplifying Boolean expressions. It is a graphical technique called **Karnaugh mapping**. After wrestling with the laws of Boolean algebra, you will really appreciate Karnaugh mapping. Because Karnaugh mapping simplifies the simplification process, I promise not to dwell too much on the laws of Boolean algebra.

Simplification Using Boolean Algebra

Let's begin by simply providing a list of several of the more common laws of Boolean algebra. The laws listed in Table 2-2 are the ones that you will use in this course.

Table 2-2: Some Common Laws of Boolean Algebra

1. $\overline{\overline{A}} = A$
2. $A+1 = 1$
3. $A \bullet 1 = A$
4. $A+A = A$
5. $A+\overline{A} = 1$
6. $A \bullet \overline{A} = 0$
7. $A \bullet 0 = 0$
8. $A \bullet A = A$
9. $(A+B) \bullet (A+C) = (A+B)(A+C) = A+BC$
10. $A \bullet (B+C) = A(B+C) = AB+AC$
11. $A+\overline{A}B = A+B$
12. $\overline{A+B} = \overline{A}\,\overline{B}$ (DeMorgan's 1'st)
13. $\overline{AB} = \overline{A}+\overline{B}$ (DeMorgan's 2'd)
14. $A\overline{B}+\overline{A}B = A \oplus B$

You have already learned a few of the laws listed in Table 2-2, like DeMorgan's Theorems. In fact, you have already proven these particular laws. Do you recall how they were proven? You were right if you said, "By truth tables."

Any law, or relationship, in Boolean algebra can be proven by truth tables. Let's just prove one of the laws in Table 2-2 to recall the procedure. How about Law 10: $A(B+C) = AB+AC$.

Suppose you let $y_1 = A(B+C)$, and $y_2 = AB+AC$. Now, you could get the truth table for y_1 and y_2 by substituting all the possible logic combinations of A, B, and C using brute force as you have done in the past. However, there must be an easier way!

Look at the equation for y_1. Notice that y_1 cannot possibly be high if A is low, since A is being ANDed with the quantity $(B+C)$. For that same reason, y_1 cannot be high when both B and C are low. This immediately results in the first five entries in the following truth table.

Now, all you need is the last three entries. Again, looking at the equation, the only time that y_1 can be high is when A is high AND B OR C are high. This is the case for each of the last three table entries.

ABC	y_1
000	0
001	0
010	0
011	0
100	0
101	1
110	1
111	1

In the same way, when you observe the equation for y_2, you see that y_2 cannot possibly be high when A is low, because of the ANDing operations. Then, if A is high, either B must be high or C must be high to make y_2 high. Thus, the above common sense reasoning dictates that the truth table must be:

ABC	y_2
000	0
001	0
010	0
011	0
100	0
101	1
110	1
111	1

Since the two truth tables are identical, Law 10 is proven. Notice how the truth tables were deduced by just knowing the basic logic operations and using a little common sense. You were forced to look at the overall equation rather than the individual 1's and 0's within the equation. I refer to this common sense reasoning approach as "looking at the forest before the trees." Many times you get a much better perspective and can easily determine what's going on in the equation by standing back and looking at the overall picture, rather than immediately plunging-in with 1's and 0's. You might want to try to prove some of the other Boolean laws using this approach.

At this time, let's see how the preceding laws can be applied to simplify Boolean expressions. As an example, consider the voter circuit problem that you worked on in the last section. When you left the problem, you had used the truth table to develop the Boolean expression:

$$y = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Like conventional algebra, no one thing will work in every case when trying to reduce an equation. So, where do you start? Many times, the place to start is to look for variables which are common to two or more terms in the equation. These common variables can then be "factored" out of their respective terms. (Which law of Boolean algebra allows you to do this?) Observing the voter circuit equation, you find that "BC" is common to the first and last terms of the equation. Thus, by Law 10, you can rewrite the equation as:

$$y = BC(\overline{A} + A) + A\overline{B}C + AB\overline{C}$$

Now, applying Law 5, you get:

$$y = (BC \cdot 1) + A\overline{B}C + AB\overline{C}$$

The next step is to apply Law 3 to get:

$$y = BC + A\overline{B}C + AB\overline{C}$$

Again, look for common terms. Factoring a "C" out of the first two terms gives you:

$$y = C(B + A\overline{B}) + AB\overline{C}$$

Continuing in this manner, applying the laws of Boolean algebra whenever possible, you should get the following sequence of operations:

$$\begin{aligned}
y &= C(A+B)+AB\overline{C} && \text{by Law 11}\\
y &= AC+BC+AB\overline{C} && \text{by Law 10}\\
y &= AC+B(C+A\overline{C}) && \text{by Law 10}\\
y &= AC+B(C+A) && \text{by Law 11}\\
y &= AC+BC+AB && \text{by Law 10}
\end{aligned}$$

You should verify each of the above steps in order to understand how the given laws were applied. Notice that the original equation has been reduced from four 3-bit operations to three 2-bit operations. However, the reduction process is quite tedious. Of course, other reductions are possible, especially if you use the exclusive OR operation in Law 14. Try it!

This brings out a problem with simplification using Boolean algebra — you never really know if your reduced equation is the simplest equation unless you try all the reduction possibilities.

Example 2-3

In Example 2-2, you obtained a Boolean expression for the "e" segment of a common anode 7-segment LED decoder circuit. Simplify this equation using the laws of Boolean algebra provided in Table 2-2.

The equation obtained from the truth table in Example 2-2 was:

$$\begin{aligned}
e &= \overline{A}\overline{B}\overline{C}D+\overline{A}\overline{B}CD+\overline{A}B\overline{C}\overline{D}+\overline{A}B\overline{C}D+\overline{A}BCD+A\overline{B}\overline{C}D\\
&= \overline{A}\overline{B}D(\overline{C}+C)+\overline{A}BD(\overline{C}+C)+\overline{A}B\overline{C}\overline{D}+A\overline{B}\overline{C}D && \text{by Law 10}\\
&= \overline{A}\overline{B}D(1)+\overline{A}BD(1)+\overline{A}B\overline{C}\overline{D}+A\overline{B}\overline{C}D && \text{by Law 5}\\
&= \overline{A}\overline{B}D+\overline{A}BD+\overline{A}B\overline{C}\overline{D}+A\overline{B}\overline{C}D && \text{by Law 3}\\
&= \overline{A}\overline{B}D+\overline{A}B(D+\overline{C}\overline{D})+A\overline{B}\overline{C}D && \text{by Law 10}\\
&= \overline{A}\overline{B}D+\overline{A}B(D+\overline{C})+A\overline{B}\overline{C}D && \text{by Law 11}\\
&= \overline{B}D(\overline{A}+A\overline{C})+\overline{A}B(D+\overline{C}) && \text{by Law 10}\\
&= \overline{B}D(\overline{A}+\overline{C})+\overline{A}B(D+\overline{C}) && \text{by Law 11}\\
&= \overline{B}D\overline{A}+\overline{B}D\overline{C}+\overline{A}BD+\overline{A}B\overline{C} && \text{by Law 10}\\
&= \overline{A}\overline{B}D+\overline{A}BD+\overline{A}B\overline{C}+\overline{B}\overline{C}D && \text{rearrange terms}\\
&= \overline{A}D(\overline{B}+B)+\overline{A}B\overline{C}+\overline{B}\overline{C}D && \text{by Law 10}\\
&= \overline{A}D(1)+\overline{A}B\overline{C}+\overline{B}\overline{C}D && \text{by Law 5}\\
&= \overline{A}D+\overline{A}B\overline{C}+\overline{B}\overline{C}D && \text{by Law 3}\\
&= \overline{A}D+\overline{C}(\overline{A}B+\overline{B}D) && \text{by Law 10}
\end{aligned}$$

Again, as you can see from the above example, simplification using Boolean algebra is indeed a tedious job. Furthermore, when you think you are finished, you might not have the simplest equation. There must be an easier, more precise way to reduce Boolean equations! There is, and it is a graphical technique called Karnaugh mapping. Please don't be too disappointed if I now go on to Karnaugh mapping in lieu of Boolean algebra.

Karnaugh Mapping

Karnaugh mapping is preferred over Boolean algebra because it makes the simplification process faster, easier, and more effective. In fact, Karnaugh mapping completely eliminates the need for using Boolean algebra, and it allows you to translate the truth table directly into a simplified equation. In this section, you will learn how to perform Karnaugh mapping for problems with two, three, and four input variables. You will find that this is adequate to solve most application tasks.

TWO INPUT VARIABLES

The first thing you must do when performing a Karnaugh map, or K-map, is to construct the grid that will be used during the mapping process. A two-variable grid is shown in Figure 2-10a. Here, the two input variables are A and B. The possible logic states for A are listed in the left column, while those for B are listed in the top row. The intersection of a row and a column in the grid is called a **cell**. Once the grid is set-up, the process involves three operations: plotting, grouping, and reading. The best way to teach you how to plot, group, and read is to look at some examples.

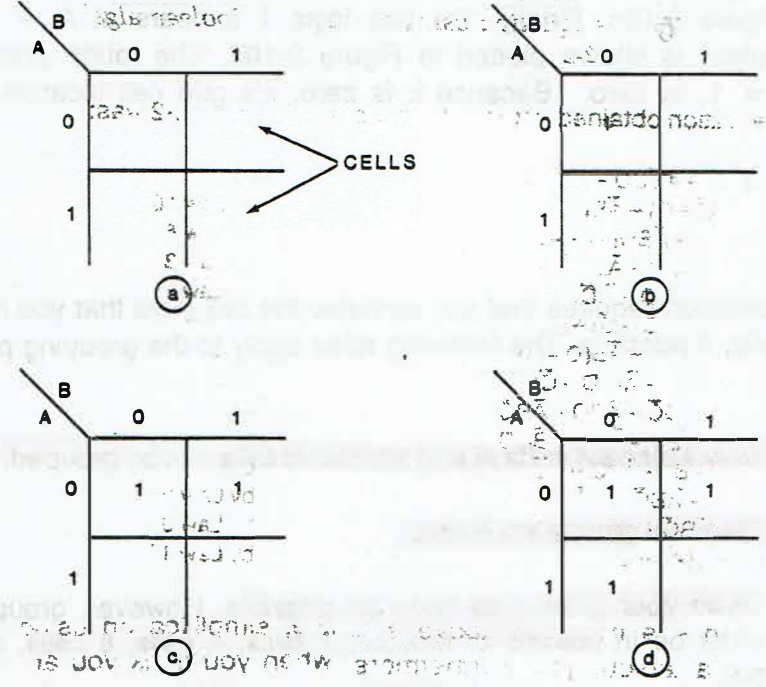


Figure 2-10

The two-variable grid a is used to plot the logic 1 outputs of the truth table in b, c, and d.

Plotting

Suppose you have arrived at the following truth table:

A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

To plot a truth table on a K-map grid, you must transfer the logic 1 outputs of the truth table to the grid cells. The first logic 1 output in the preceding truth table appears for an input of A = 0 and B = 0. To plot this output, place a 1 in the grid cell where the A = 0 column and B = 0 row intersect as shown in Figure 2-10b. Notice that this is the same way that you plot points on an x-y coordinate system in algebra class.

The next logic 1 output in the truth table appears at A = 0 and B = 1. Again, place a 1 in the grid cell where the A = 0 column and B = 1 row intersect, as shown in Figure 2-10c. Finally, the last logic 1 appears at A = 1 and B = 0. This output is shown plotted in Figure 2-10d. The fourth possibility, A = 1 and B = 1, is zero. Because it is zero, it's grid cell location is not plotted.

Grouping

The grouping operation requires that you combine the cell plots that you marked on the K-map grid, if possible. The following rules apply to the grouping process (study them!):

1. Only **adjacent** vertical and horizontal cells can be grouped.
2. Diagonal groups are illegal.
3. Make your groups as large as possible. However, group sizes must be in powers of two, i.e. 2 cells, 4 cells, 8 cells, and so on.

4. All plots must be grouped, if possible, using the above rules.

5. A given plot may be used to form more than one group, thereby creating overlapping groups.

6. If no grouping is possible, the equation is in its simplest form and cannot be reduced further.

Now, apply the above rules to the final plot in Figure 1-10d. How many groups do you see? You were right if you said, "two." First, the two adjacent vertical plots can be grouped as shown in Figure 2-11a. Then, the two adjacent horizontal plots can be grouped as shown in Figure 2-11b. Notice that the two groups overlap (share a plot). As a result, you end up with two groups of two. Let's label the vertical group, group 1, and the horizontal group, group 2, as shown in the figure. The grouping shown in Figure 2-11C will not work, since diagonal cell groups are illegal.

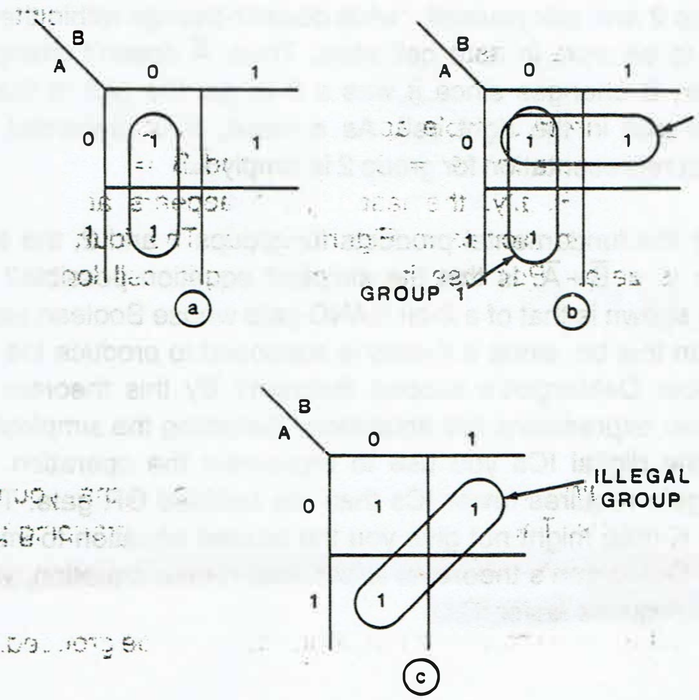


Figure 2-11

The first group of two is a vertical group a, and the second group of two is a horizontal group which overlaps the first group b. Diagonal groups are illegal c.

Reading the Groups

The final step in performing a K-map is to read the groups and generate the simplified equation. You must read **each** group within the K-map and generate a fundamental product term for each group. The fundamental product terms generated by each group are then ORed together to produce the final sum-of-products equation.

To read a group, you must go from one cell within the group to the next, asking yourself: "what doesn't change within the group?" For example, consider group 1 in Figure 2-11b. Notice that the input variable A changes from one cell to the next. The variable A had to be a 0 for the top cell plot and a 1 for the bottom cell plot. Since A has changed within the group, it is ignored. Now look at what the input variable B does within group 1. For the top cell plot, B had to be a 0. Also, B had to be a 0 in the bottom cell plot. Thus, B **doesn't change** and remains a 0 within the group. Since B didn't change and remained a 0 within the group, the fundamental product representation for group 1 is $\overline{B}$.

THe item which
Does not change
within a group -
Becomes the New
group

Next, consider group 2 and ask yourself, "what doesn't change within the group?" Notice that A had to be zero in both cell plots. Thus, $\overline{A}$ doesn't change within the group. However, B changes since it was a 0 to get the plot in the left cell and a 1 to get the plot in the right cell. As a result, B is discarded and the fundamental product representation for group 2 is simply $\overline{A}$.

Combining (ORing) the fundamental products for groups 1 and 2, the simplified equation becomes: $y = \overline{B}+\overline{A}$. Is this the simplest equation possible? The associated truth table shown is that of a 2-bit NAND gate whose Boolean expression is: $y = \overline{AB}$. How can this be, since a K-map is supposed to produce the simplest equation? Remember DeMorgan's second theorem? By this theorem (law 13 in Table 2-2), the two expressions are equivalent. Selecting the simplest expression depends on the digital ICs you use to implement the operation. In most cases, the **NAND** gate requires fewer ICs than the bubbled OR gate. This goes to show you that a K-map might not give you the easiest equation to implement. By applying one of DeMorgan's theorems to the final K-map equation, you might get an equation that requires fewer ICs.

Example 2-4

Use a K-map to get the simplified equation for the following truth table:

A	B	y
0	0	0
0	1	1
1	0	1
1	1	0

Here is an example where a little common sense is appropriate. Looking at the truth table, you see a simple 2-bit XOR gate. Thus, the required equation becomes:

$$y = A \oplus B$$

However, suppose that you didn't recognize the truth table and proceeded with a K-map. The resulting K-map plot is shown in Figure 2-12. No groups are possible, since diagonal groups are illegal. You could say that the K-map reduces to two groups of one cell each. This means that the equation can not be simplified using a K-map.

Reading the two groups of one, the fundamental product that produced the plot in the lower left cell is $A\overline{B}$. The fundamental product that produced the plot in the upper right cell is $\overline{A}B$. The equation then becomes $A\overline{B} + \overline{A}B$. Thus, the K-map gave you $A\overline{B} + \overline{A}B$, and Law 14 in Table 2-2 reduces the equation to $A \oplus B$.

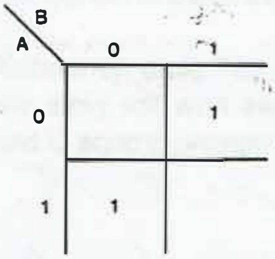


Figure 2-12

If a K-map plot cannot be grouped, the Boolean expression is in its simplest form.

The lesson to be learned here is that K-maps do not provide you with XOR operations in the resulting expression. Consequently, you must always look for the terms $A\overline{B} + \overline{A}B$ in the K-map equation and reduce them to $A \oplus B$.

THREE INPUT VARIABLES

Two variable maps are rather trivial, since you can usually apply a little common sense to the truth table and come up with the simplest possible equation. On the other hand, three and four variable maps are more challenging.

To demonstrate a three-variable map, recall the truth table that we developed for the voter circuit problem:

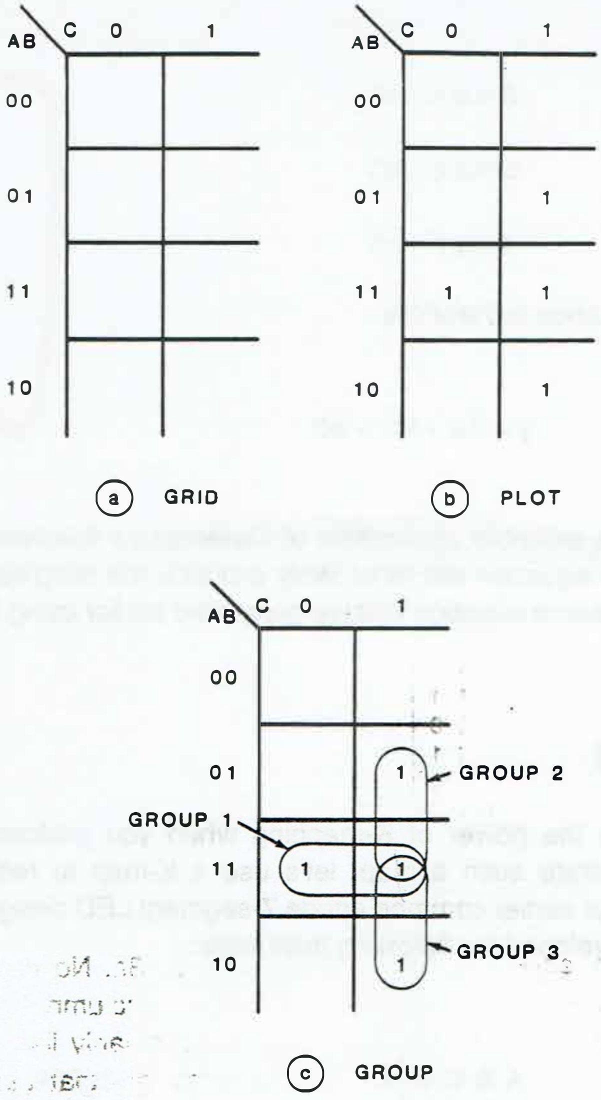
A	B	C	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The structure for a three variable grid is shown in Figure 2-13a. Notice that all possible combinations of the A and B inputs are listed in the left column. However, notice the order of the listing. This is not how you might ordinarily list the four logic combinations of two inputs; however, it is very important that this order be maintained. If not, the K-map will not work. Finally, the two logic possibilities for the third variable, C, are listed in the top row.

Figure 2-13b shows the cell plots generated by the logic 1 outputs in the truth table. Figure 2-13c shows how the plots should be grouped. Observe that the four plots form three overlapping groups of two.

IN tHe MAP Notice THAT 1's and 0's DoNoT AlTeRnATe THe 1's 0's FoR A ARe together AN Also FoR B.

0 0
0 1
1 1
1 0



(a) GRID

(b) PLOT

(c) GROUP

Figure 2-13

A three-variable K-map for the voter circuit.

Reading the groups using the labeling shown in the figure, the respective fundamental product terms are:

Group 1: AB

Group 2: BC

Group 3: AC

The resulting simplified equation is therefore:

$$y = AB + BC + AC$$

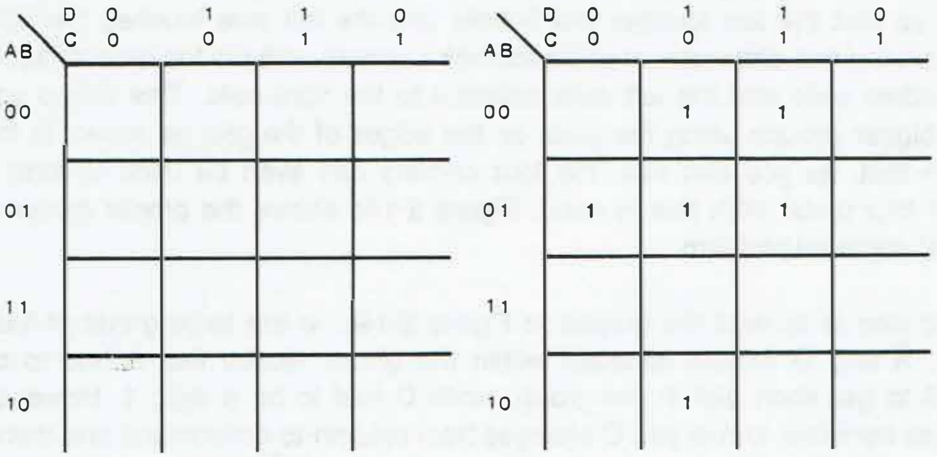
Since you don't observe a possible application of DeMorgan's theorems or an XOR operation, the above equation will most likely produce the simplest circuit. You will find that this is the same equation that we generated earlier using Boolean algebra.

FOUR INPUT VARIABLES

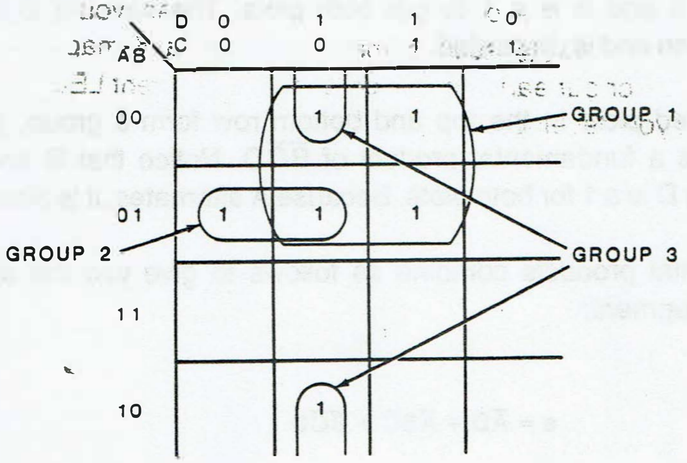
You can really appreciate the power of K-mapping when you perform a four variable map. To demonstrate such a map, let's use a K-map to reduce the "e" segment equation for our earlier common anode 7-segment LED design example. In Example 2-2 you developed the following truth table:

A B C D	e
0 0 0 0	0
0 0 0 1	1
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	1
1 0 0 0	0
1 0 0 1	1

The grid for a four variable map is shown in Figure 2-14a. The A and B input combinations are listed in the left column and the C and D input combinations are listed in the top row. Again, notice the order of the input listings. This order must be followed or the K-map will not work. Figure 2-14b shows the result when each of the logic 1 outputs from the truth table are plotted.



ⓐ GRID ⓑ PLOT



ⓒ GROUP

Figure 2-14

A four variable grid **a**, plot **b**, and grouping **c** for the "e" segment LED problem.

Now, with three and four variable maps, the grouping can get a little tricky because there are several unique possibilities. Remember that you must form the largest groups possible and the size of each group must be some power of 2. Figure 2-15 illustrates several possible ways that groups can be formed. The groups of four and eight cells that are completely circled in Figure 2-15 should be self-explanatory. Each group consists of four or eight adjacent plots. However, notice the plots that are only half-circled. These can also be used to form groups by thinking of the K-map grid as a sphere. In other words, you can mentally roll the grid so that the top touches the bottom and the left side touches the right side. Then, the two-dimensional grid becomes a sphere with the top cells adjacent to the bottom cells and the left cells adjacent to the right cells. This allows you to form bigger groups using the plots on the edges of the grid as shown in the figure. In fact, as you can see, the four corners can even be used to form a group of four plots. With this in mind, Figure 2-14c shows the proper grouping for the "e" segment problem.

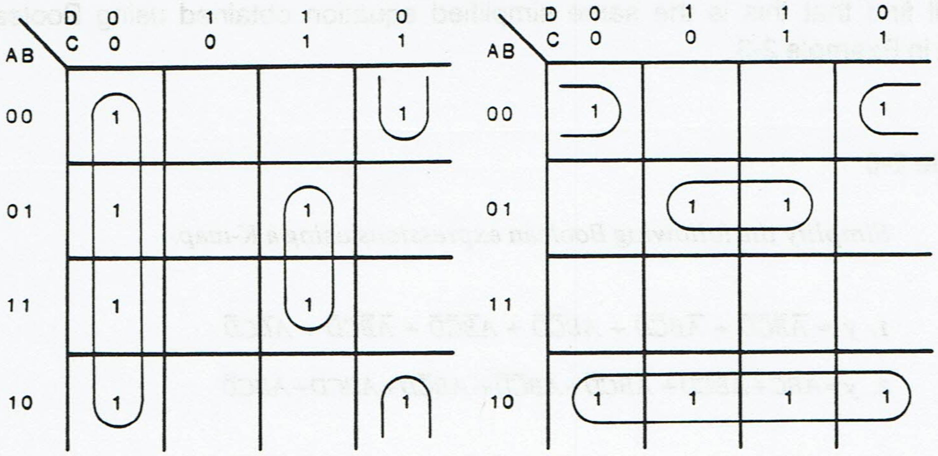
The next step is to read the groups in Figure 2-14c. In the large group of four, group 1, $\overline{A}$ and D remain constant within the group. Notice that A had to be a logic 0 to get each plot in the group, while D had to be a logic 1. However, B changes from row-to-row and C changes from column-to-column and are, therefore, discarded. Thus, the resulting fundamental product is $\overline{A}D$.

The horizontal group of two, group 2, produces a fundamental product of $\overline{A}B\overline{C}$. Here, A and C are 0 and B is a 1 to get both plots. The variable D changes from column-to-column and is discarded.

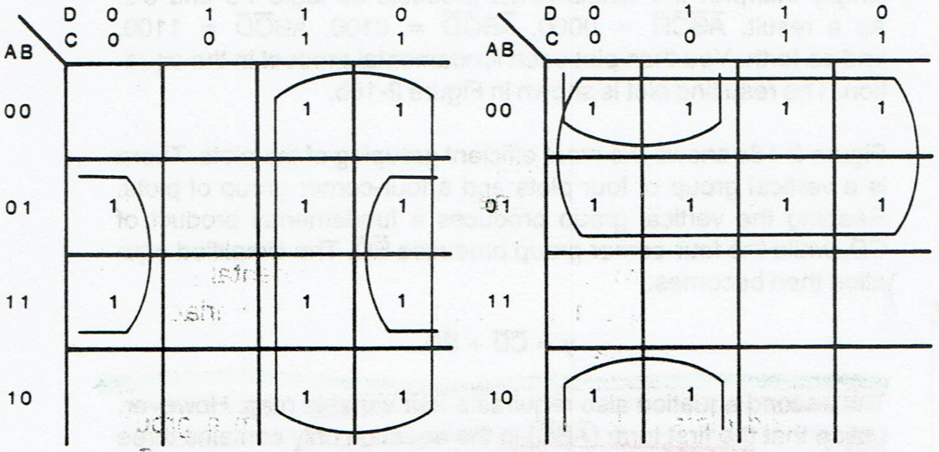
Finally, the half-circled plots in the top and bottom row form a group, group 3, of two that produces a fundamental product of $\overline{B}\overline{C}D$. Notice that B and C are 0 for both plots, while D is a 1 for both plots. Because A alternates, it is discarded.

The three fundamental products combine as follows to give you the simplified equation for the "e" segment:

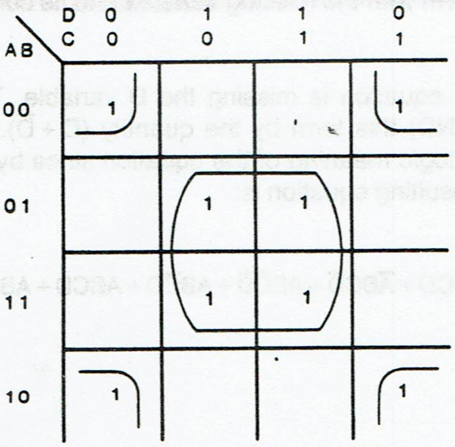
$$e = \overline{A}D + \overline{A}B\overline{C} + \overline{B}\overline{C}D$$



ⓐ COLUMN GROUPS ⓑ ROW GROUPS



ⓒ RETANGULAR GROUPS ⓓ RECTANGULAR GROUPS



ⓔ RECTANGULAR GROUPS

Figure 2-15
There are several possible ways to form groups in three- or four-variable K-maps.

You will find that this is the same simplified equation obtained using Boolean algebra in Example 2-3.

Example 2-5

Simplify the following Boolean expressions using a K-map.

1. $y = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}\overline{D} + AB\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + A\overline{B}C\overline{D}$

2. $y = \overline{A}B\overline{C} + \overline{A}BCD + \overline{A}BC\overline{D} + AB\overline{C}\overline{D} + AB\overline{C}D + ABCD + ABC\overline{D}$

The first equation requires the four variable grid shown in Figure 2-16a. To plot a Boolean equation, rather than a truth table, you simply interpret the fundamental products as logic 1's and 0's. As a result, $\overline{A}\overline{B}\overline{C}\overline{D}$ = 0000, $\overline{A}B\overline{C}\overline{D}$ = 0100, $AB\overline{C}\overline{D}$ = 1100, and so forth. You then plot each fundamental product in the equation. The resulting plot is shown in Figure 2-16b.

Figure 2-16c shows the most efficient grouping of the plots. There is a vertical group of four plots and a four-corner group of plots. Reading the vertical group produces a fundamental product of $\overline{C}\overline{D}$, while the four-corner group produces $\overline{B}\overline{D}$. The simplified equation then becomes:

$$y = \overline{C}\overline{D} + \overline{B}\overline{D}$$

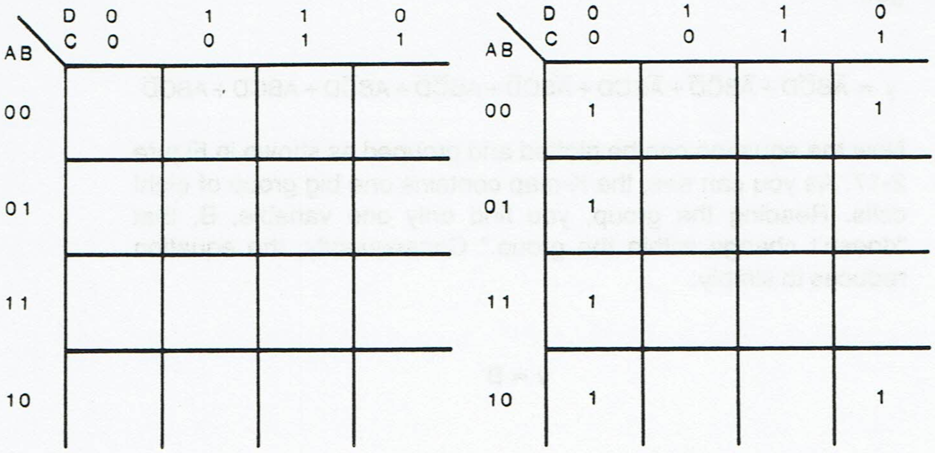
If all groups don't Have The same # of Terms (variables)

*

The second equation also requires a four variable map. However, notice that the first term ($\overline{A}B\overline{C}$) in the equation only contains three variables. To perform a K-map, **all** of the terms must contain **all** of the input variables. If a given term does not, you must multiply (AND) that term with the missing variable and its complement.

The first term in this equation is missing the D variable. Thus, you must multiply (AND) this term by the quantity $(D+\overline{D})$. This does not change the logic meaning of the equation since by Law 5, $(D+\overline{D}) = 1$. The resulting equation is:

$$y = \overline{A}B\overline{C}(D+\overline{D}) + \overline{A}BCD + \overline{A}BC\overline{D} + AB\overline{C}\overline{D} + AB\overline{C}D + ABCD + ABC\overline{D}$$

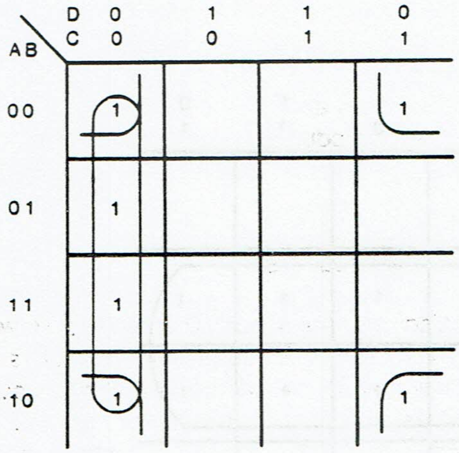


AB \ D C	0 0	1 0	1 1	0 1
00				
01				
11				
10				

ⓐ GRID

AB \ D C	0 0	1 0	1 1	0 1
00	1			1
01	1			
11	1			
10	1			1

ⓑ PLOT



AB \ D C	0 0	1 0	1 1	0 1
00	1			1
01	1			
11	1			
10	1			1

ⓒ GROUP

$y = \overline{C}\overline{D} + \overline{B}\overline{D}$

ⓓ READ

Figure 2-16

The K-map for Example 2-5a.

Multiplying (actually ANDing) through by the $(D+\overline{D})$ quantity you get:

$$y = \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D} + \overline{A}BCD + \overline{A}BC\overline{D} + AB\overline{C}\overline{D} + AB\overline{C}D + ABCD + ABC\overline{D}$$

Now the equation can be plotted and grouped as shown in Figure 2-17. As you can see, the K-map contains one big group of eight cells. Reading the group, you find only one variable, B, that "doesn't change within the group." Consequently, the equation reduces to simply:

$$y = B$$

Try doing that as easily with Boolean algebra!

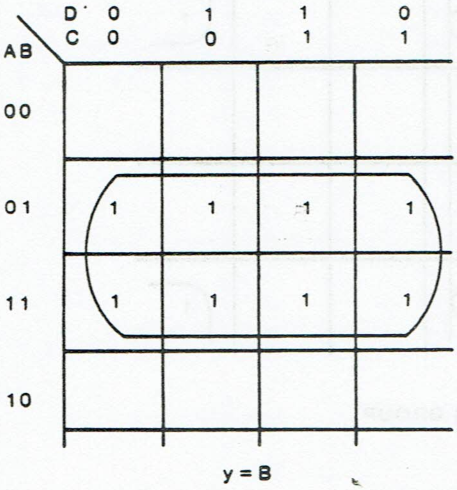


Figure 2-17

The K-map for Example 2-5b.

DON'T CARES

The last thing that you need to know about K-mapping is how to use don't cares. A **don't care** condition applies to the input variables of a problem. This condition exists when a given logic combination of the inputs is illegal or will not occur according to the application task. The easiest way to understand a don't care condition is to look at an example.

Consider the standard 4-bit BCD code. By definition, the code uses 4-bit binary values, 0000 through 1001, to represent the decimal digits 0 through 9. However, a 4-bit code can range from 0000 to 1111. What happens to the extra six codes, 1010 through 1111, in BCD? By definition, they are illegal codes and should not occur in a BCD system. As a result, they qualify as don't care conditions in a design problem.

Now, let's go back to our 7-segment LED design problem. The inputs to the circuit generate the 4-bit BCD code. Up to this point, the associated truth table for the "e" segment has only shown the inputs ranging from 0000 through 1001, since any value above 1001 is illegal and should not occur in a BCD system. By definition, any input value above 1001 constitutes a don't care condition. So, let's list these values in the truth table and place an **X** in the output column to indicate the don't care condition. The resulting truth table is shown below.

A	B	C	D	e
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Illegal BCD Codes

Don't Care Conditions

The don't care conditions are labeled with an X to indicate that the output for these conditions can be treated as either a 1 or a 0, we simply "don't care," since the conditions should never occur in the circuit anyway. However, if you treat the don't care conditions as producing logic 1 outputs, you can plot them in the K-map and use them to form larger groups. Larger groups result in simpler equations.

Figure 2-18a shows the logic 1 outputs of the truth table plotted as 1's. In addition, the don't care conditions of the truth table are plotted using X's. In Figure 2-18b, you see that larger groups can be formed by using the don't cares. The equation for the "e" segment now reduces to simply

$$e = B\overline{C} + D$$

from our earlier

$$e = \overline{A}D + \overline{A}B\overline{C} + B\overline{C}D$$

The idea is to plot the don't care conditions and use them, if you can, to make larger groups. **Do not** use them if they are not needed. In other words, don't try to group all the X's! They are only used in conjunction with the real "1's" to generate larger groups. Also, do not try to form a separate group consisting solely of don't care conditions. A group of all X's will produce an unneeded term in the final equation.

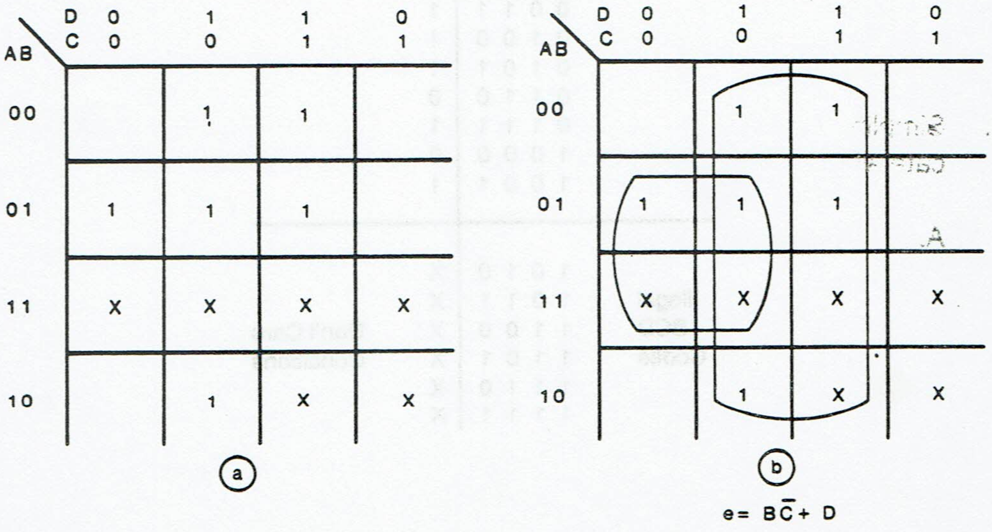


Figure 2-18

Don't cares are plotted **a** and used as needed to form larger groups **b**.

Self-Test Review

18. Explain why a Boolean expression must be simplified before it can be used to implement logic gates in a digital circuit. ______________________

____

____

____

19. Prove Boolean Law 11 in Table 2-2.

20. Simplify the following expressions using the laws of Boolean algebra. Indicate which law in Table 2-2 allows you to perform each operation.

A. $y = A\overline{B}C + \overline{A}BC$

B. $x = \overline{A}BC + ABC$

C. $f(A,B,C) = \Sigma(5,6,7)$

D. $z = \overline{(\overline{AB})(\overline{AB})}$

21. Given the following truth table, write the Boolean expressions for j and k:

A	B	C	D	j	k
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	1	X
0	1	1	1	1	1
1	0	0	0	1	1
1	0	0	1	1	0
1	0	1	0	X	X
1	0	1	1	X	0
1	1	0	0	X	0
1	1	0	1	X	X
1	1	1	0	X	0
1	1	1	1	X	X

j = ____

k = ____

22. Perform a K-map for j and for k in question 21.

23. Write the simplified Boolean expression for each of your K-maps in question 22.

j = ______________________________

k = ______________________________

24. How many and what size AND/OR gates must be used to implement j and k in question 23?

j = ______________________________

k = ______________________________

25. Simplify the following equation:

$$z = ABCD + AB\overline{C}D + ABC\overline{D} + A\overline{B}\overline{C}D + AB\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B}C$$

Answers

18. Simplifying the expression simplifies the logic gate circuit design. A simpler design results in fewer ICs, smaller printed circuit boards, and greater reliability.

19. Solution follows.

20. Solution follows.

21. $j = \overline{A}B\overline{C}\overline{D} + \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D$

$k = \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + A\overline{B}\overline{C}\overline{D}$

22. Refer to Figure 2-19.

23. $j = A + \overline{C}\overline{D} + BC$

$k = BD + \overline{B}\overline{D}$

24j. Two 2-bit AND gates and a 3-bit OR gate.

24k. Two 2-bit AND gates and a 2-bit OR gate.

25. Solution follows.

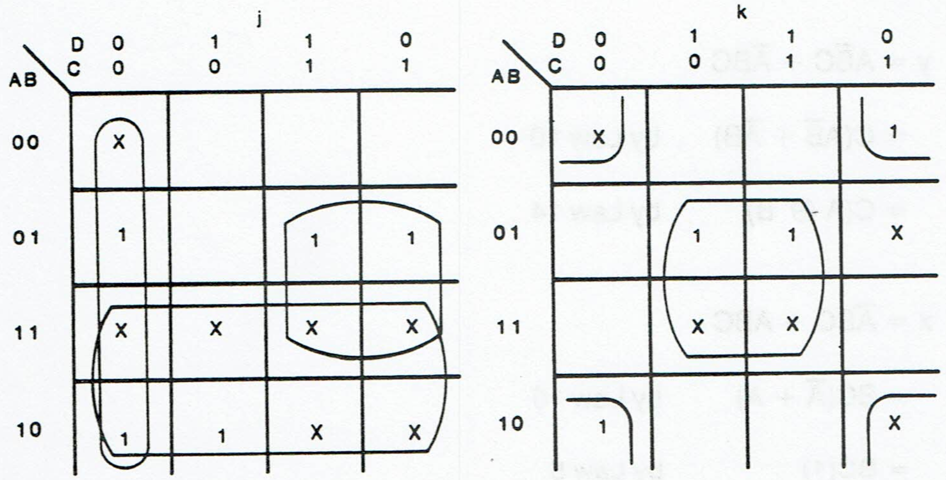


Figure 2-19

The K-maps for question 22.

The solution for questions 19, 20, and 25 follow:

19. Law 11 states that $A + \overline{A}B = A + B$. Let $y_1 = A + \overline{A}B$ and $y_2 = A + B$. Notice that y_2 is a simple 2-bit OR gate, thus its truth table is:

A	B	y_2
0	0	0
0	1	1
1	0	1
1	1	1

Looking at the $y_1 = A + \overline{A}B$ expression, you see that y_1 is high whenever A is high because of the ORing operation. In addition, y_1 is high whenever A is low AND B is high. As a result, the truth table is:

A	B	y_1
0	0	0
0	1	1
1	0	1
1	1	1

Since the truth tables are the same, Law 11 is proven.

20A. $y = A\overline{B}C + \overline{A}BC$

$= C(A\overline{B} + \overline{A}B)$ by Law 10

$= C(A \oplus B)$ by Law 14

20B. $x = \overline{A}BC + ABC$

$= BC(\overline{A} + A)$ by Law 10

$= BC(1)$ by Law 5

$= BC$ by Law 3

20C. $f(A,B,C) = \Sigma(5,6,7)$

$$= A\overline{B}C + AB\overline{C} + ABC$$

$$= A\overline{B}C + AB(\overline{C} + C) \quad \text{by Law 10}$$

$$= A\overline{B}C + AB(1) \quad \text{by Law 5}$$

$$= A\overline{B}C + AB \quad \text{by Law 3}$$

$$= A(\overline{B}C + B) \quad \text{by Law 10}$$

$$= A(C + B) \quad \text{by Law 11}$$

20D. $z = \overline{(\overline{AB})(\overline{AB})}$

$$= \overline{\overline{AB}} \quad \text{by Law 8}$$

$$= AB \quad \text{by Law 1}$$

25. You can use Boolean algebra if you want, but I prefer a K-map (see Page 2-56). First, notice that the last term of the expression does not contain the variable D. Thus, it must be ANDed with the quantity $(D + \overline{D})$. This results in the following:

$$z = ABCD + AB\overline{C}D + ABC\overline{D} + A\overline{B}\overline{C}D + AB\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B}C\,(D + \overline{D})$$

$$= ABCD + AB\overline{C}D + ABC\overline{D} + A\overline{B}\overline{C}D + AB\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D}$$

Now, the equation can be plotted and grouped as shown in Figure 2-20. Reading the single group of eight cells results in the equation:

$$z = A$$

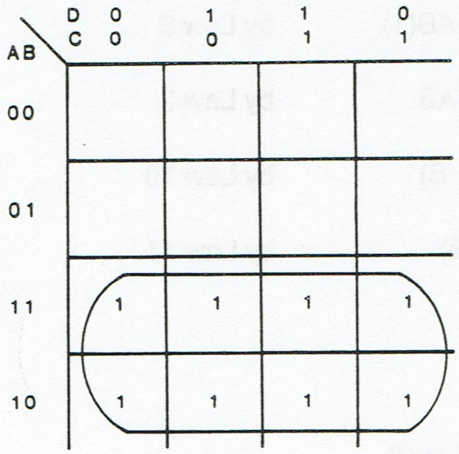


Figure 2-20

The K-map for question 25.

Before you read the next section, perform Experiment 5 in Unit 8.

SSI IMPLEMENTATION

This final section discusses the last two steps in the design algorithm: the AND/OR design and the NAND/NOR/XOR design. As you are already aware, you can construct an AND/OR circuit directly from a Boolean equation. However, such circuits are often not as economical and easy to implement as an equivalent NAND/NOR/XOR circuit. So, the last step in the design process usually requires you to convert the AND/OR design to a NAND/NOR/XOR design.

The AND/OR Design

The AND/OR design is taken directly from the simplified Boolean expression. Sketching the circuit requires you to observe the logic operations within the equation. For example, recall the simplified equation that you obtained for the voter circuit:

$$y = AB + BC + AC$$

Since the simplified equation will always be in sum-of-products form, the circuit will always consist of one or more AND gates driving an OR gate. Looking at the above equation, you see that three quantities are being ORed. They are: AB, BC, and AC. Thus, working backwards from the output, y, the circuit requires a 3-bit OR gate as shown in Figure 2-21a. Now, each of the three quantities being ORed is the result of a 2-bit ANDing operation. Therefore, the circuit requires three 2-bit AND gates as shown in Figure 2-21b. For diagramming clarity, the circuit inputs are usually shown coming from a **bus** structure as in Figure 2-21c.

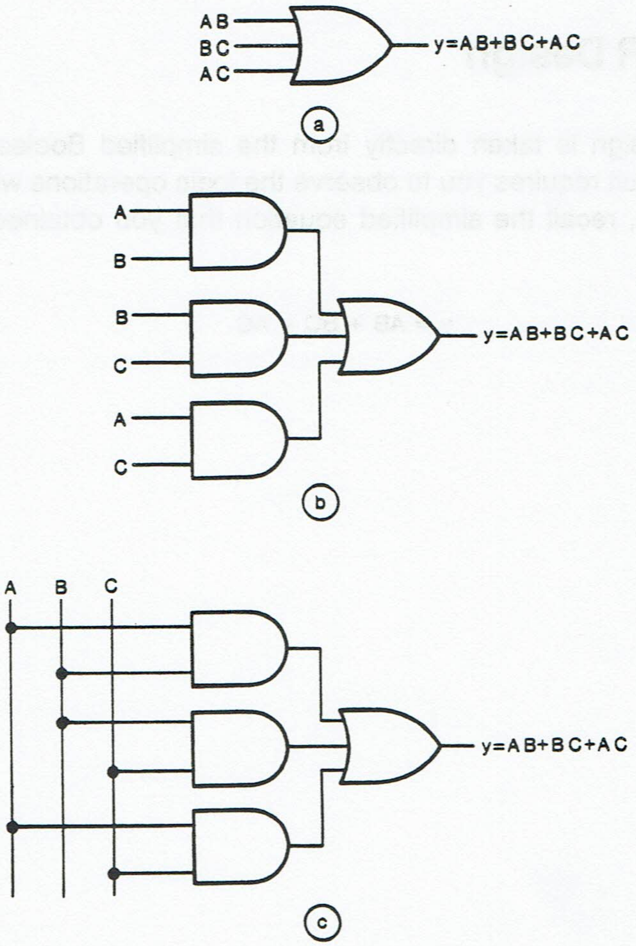


Figure 2-21

The AND/OR circuit derived from the simplified SOP expression will always consist of one or more AND gates driving an OR gate.

Example 2-6

Sketch an AND/OR circuit that can be used to illuminate the "e" segment of a 7-segment common anode LED from a BCD input.

The simplified equation that you have developed for this circuit is:

$$e = B\overline{C} + D$$

Observing the above equation, you see that two quantities are being ORed: $B\overline{C}$ and D. Thus, the circuit requires a 2-bit OR gate as shown in Figure 2-22a.

One of the inputs to the OR gate is simply D. The other input is the output of a 2-bit AND gate as shown in Figure 2-22b.

The completed circuit design is shown in Figure 2-22c. Notice that all of the bus lines are shown as positive logic. To obtain the $\overline{C}$ logic for the AND gate input, an inverter is placed between the C bus line and the gate.

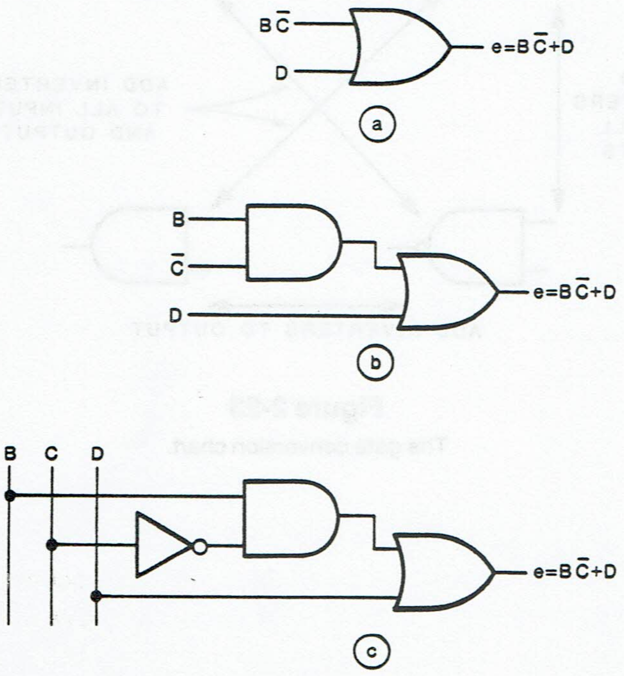


Figure 2-22

The complete AND/OR design for the "e" segment of the 7-segment LED problem.

The NAND/NOR/XOR Design

Many times the design can be built more economically using a combination of NAND, NOR, and XOR gates. In particular, the NAND gate is one of the least expensive gates in most any series of IC devices. For this reason, the AND/OR design is often converted to a circuit consisting of strictly NAND gates.

The gate conversion chart in Figure 2-23 will help you convert a AND/OR design to a NAND/NOR design. Here's how it works:

1. Substitute the desired gate for each gate in the AND/OR design.
2. Add inverters to the gate inputs and/or outputs as dictated by the chart.
3. Cancel all double inversions on a given signal path.

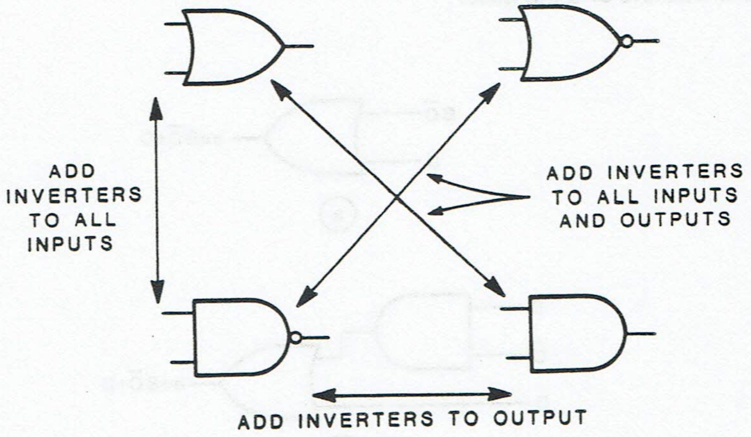
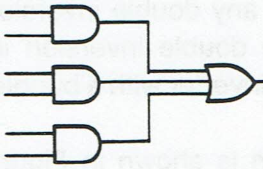

ADD INVERTERS TO ALL INPUTS
ADD INVERTERS TO ALL INPUTS AND OUTPUTS
ADD INVERTERS TO OUTPUT

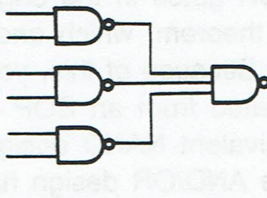
Figure 2-23

The gate conversion chart.

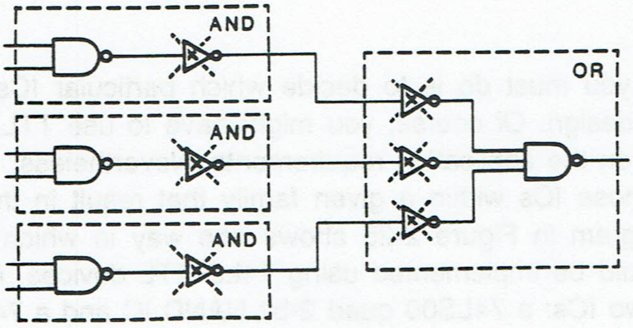
Let's suppose that you must convert the AND/OR voter circuit design to an equivalent NAND design. The AND/OR design is shown again in Figure 2-24a. The first thing you must do is substitute NAND gates directly for the AND/OR gates as shown in Figure 2-24b. But, you can't stop here because this direct substitution might not produce an equivalent circuit.



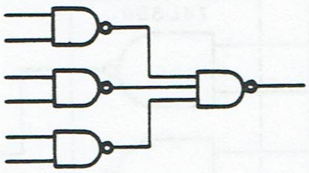
ⓐ ORIGINAL AND/OR CIRCUIT



ⓑ SUBSTITUTE DESIRED GATES



ⓒ ADD INVERTERS PER CHART AND CANCEL DOUBLE INVERSIONS



ⓓ EQUIVALENT NAND DESIGN

Figure 2-24

Gate Conversion.

Next, use the gate conversion chart to determine where inverters must be added to produce an equivalent circuit. To make a NAND gate perform like an OR gate, the chart dictates that inverters must be added to the NAND gate inputs as shown in Figure 2-24c. To make a NAND gate act like an AND gate, the chart says that an inverter must be added to the NAND gate output as shown in Figure 2-24c.

Finally, you must cancel out any double inversions as shown in the figure. This can only be done when the double inversion is the result of two inverters in series. You cannot cancel an inverter with a bubble that is part of a gate.

The equivalent NAND design is shown in Figure 2-24d. Notice that you could have just as easily ended-up with an equivalent circuit by substituting NAND gates directly for the AND/OR gates in the original design. This "coincidence" is the result of DeMorgan's theorem, which one? (Hint: a bubbled OR gate is equivalent to a NAND gate.) Because of this, you can always take an AND/OR design that has been generated from an SOP equation and substitute NAND gates directly to get an equivalent NAND design. However, be careful! Direct substitution only works if the AND/OR design has been obtained directly from a sum-of-products equation. It's still a good idea to use the conversion chart to be sure.

The last thing you must do is to decide which particular ICs you will use to implement the design. Of course, you might have to use TTL, ECL, or CMOS ICs depending on the application requirements. Nevertheless, for economy you must choose those ICs within a given family that result in the minimum chip count. The diagram in Figure 2-25 shows one way in which the NAND gate voter circuit could be implemented using 74LS TTL devices. Notice the circuit only requires two ICs: a 74LS00 quad 2-bit NAND IC and a 74LS10 triple 3-bit NAND IC.

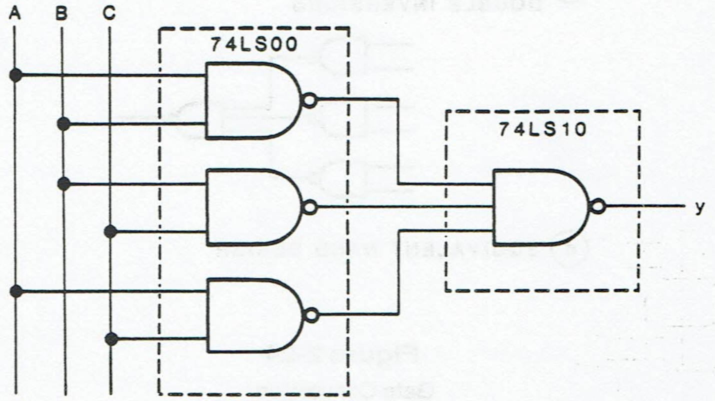


Figure 2-25
The final voter circuit can be implemented using two TTL chips — a 74LS00 and a 74LS10.

Example 2-7

Convert the NAND gate voter circuit design in Figure 2-25 to an equivalent NOR design.

The first thing you must do is to substitute NOR gates directly for the NAND gates as shown in Figure 2-26a.

Next, add inverters to the NOR gates as dictated by the gate conversion chart in Figure 2-23. The chart says, when converting a NAND to a NOR, add inverters to all inputs and outputs. The result of this operation is shown in Figure 2-26b.

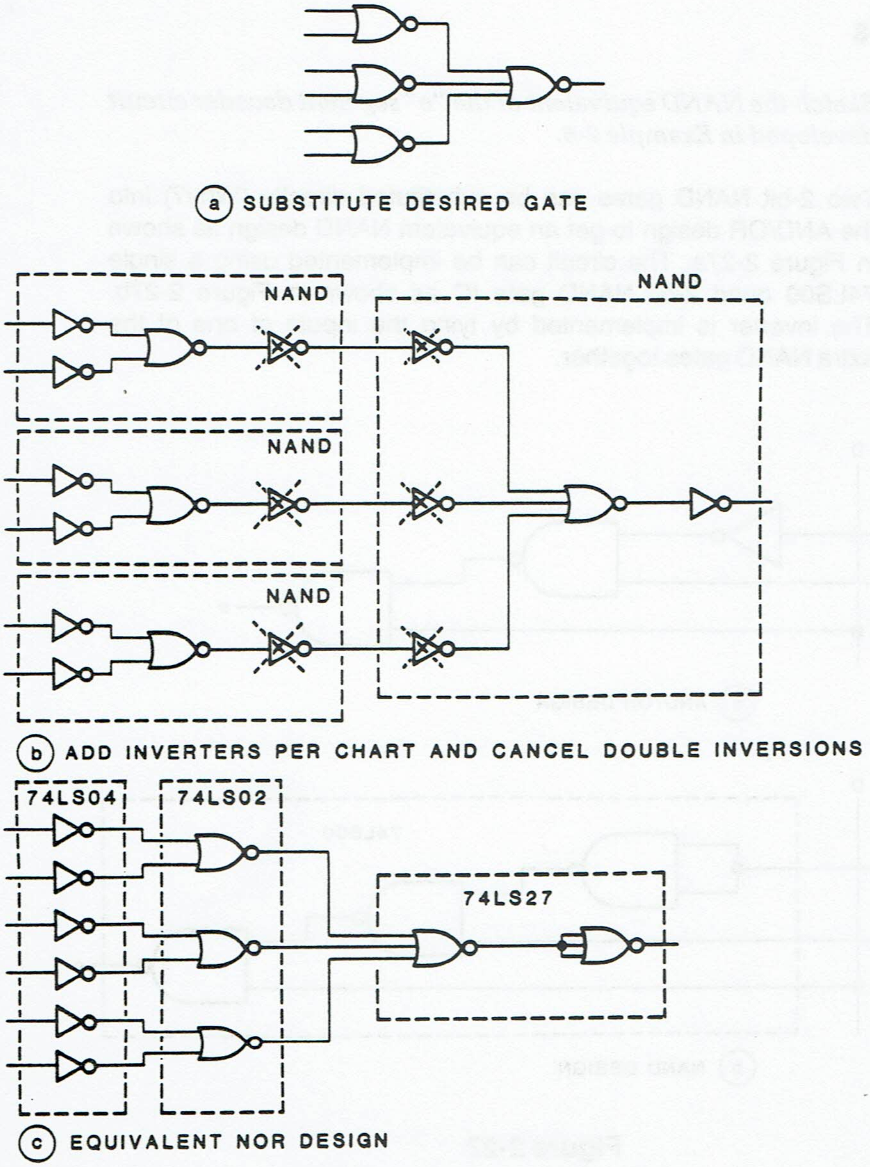

ⓐ SUBSTITUTE DESIRED GATE
NAND
NAND
NAND
NAND
ⓑ ADD INVERTERS PER CHART AND CANCEL DOUBLE INVERSIONS
74LS04
74LS02
74LS27
ⓒ EQUIVALENT NOR DESIGN

Figure 2-26

A NOR design for the voter circuit (Example 2-7).

Finally, cancel any double inversions to get the completed circuit shown in Figure 2-26c. As the figure shows, this circuit can be implemented using three chips: a 74LS04 hex inverter, a 74LS02 quad 2-bit NOR, and a 74LS27 triple 3-bit NOR.

How can this be, since the circuit requires seven inverters and a 74LS04 only contains six inverters? The "trick" is to use the extra 3-bit NOR gate in the 74LS27 chip as the seventh inverter. Recall from Unit 1 that tying the inputs of a NAND or a NOR gate together creates an inverter. This little "trick" has saved an IC. A substantial economical savings would result for large production quantities.

Example 2-8

Sketch the NAND equivalent of the "e" segment decoder circuit developed in Example 2-6.

Two 2-bit NAND gates can be substituted directly (Why?) into the AND/OR design to get an equivalent NAND design as shown in Figure 2-27a. The circuit can be implemented using a single 74LS00 quad 2-bit NAND gate IC as shown in Figure 2-27b. The inverter is implemented by tying the inputs of one of the extra NAND gates together.

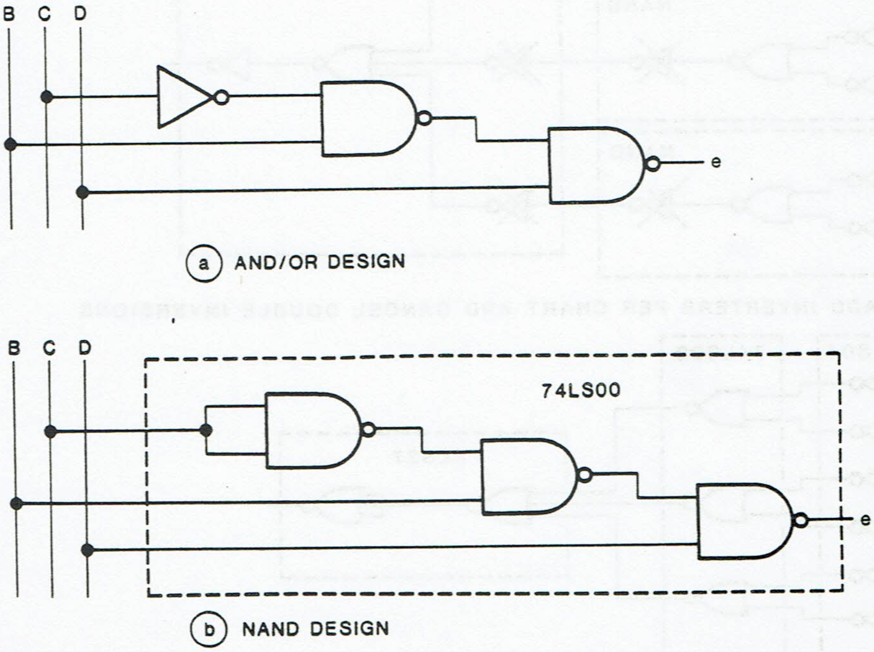


Figure 2-27

The equivalent NAND design for the "e" segment problem can be implemented using a single 74LS00 IC.

Self-Test Review

26. Sketch AND/OR circuits for the following:

$y = \overline{A}B + AC$

$j = A + \overline{C}\overline{D} + BC$

$k = BD + \overline{B}\overline{D}$

27. Why is an AND/OR design usually converted to a NAND/NOR/XOR design?

____

____

28. Convert the AND/OR circuits for y, j, and k in question 26 to NAND circuits.

29. Convert the y, j, and k circuits in question 26 to NOR circuits.

Table 2-3: ICs To Implement The Circuit Designs

Device Number	Functional Description
74LS00	Quad 2-bit NAND
74LS02	Quad 2-bit NOR
74LS04	Hex Inverter
74LS10	Triple 3-bit NAND
74LS27	Triple 3-bit NOR

30. List the TTL ICs that must be used to implement the NAND designs for y, j, and k in question 28. Use those ICs that produce a minimum chip count. Refer to the list of ICs in Table 2-3 for your choices.

y. ______________________________

j. ______________________________

k. ______________________________

31. List the TTL ICs that must be used to implement the NOR designs for y, j, and k in question 29. Use those ICs that produce a minimum chip count. Refer to the list of ICs in Table 2-3 for your choices.

y. ______________________________

j. ______________________________

k. ______________________________

Answers

26. See Figure 2-28.

27. These designs are more economical, available, and easier to implement.

28. See Figure 2-29. Notice how the bus structure is used to input data to the three circuits.

29. See Figure 2-30.

30y. One 74LS00 quad 2-bit NAND IC. (The inverter for $\overline{A}$ is made by tying the inputs of the fourth NAND gate together.)

30j. One 74LS00 quad 2-bit NAND IC and one 74LS10 triple 3-bit NAND IC. (The two inverters are made from extra NAND gates on these ICs.)

30k. One 74LS00 quad 2-bit NAND IC, and one 74LS04 hex inverter IC.

31y. One 74LS02 quad 2-bit NOR IC and a 74LS04 hex inverter IC.

31j. One 74LS02 quad 2-bit NOR IC and a 74LS27 triple 3-bit NOR IC. (The three inverters are made from the extra NOR gates on these ICs.)

31k. One 74LS02 quad 2-bit NOR IC and a 74LS04 hex inverter IC.

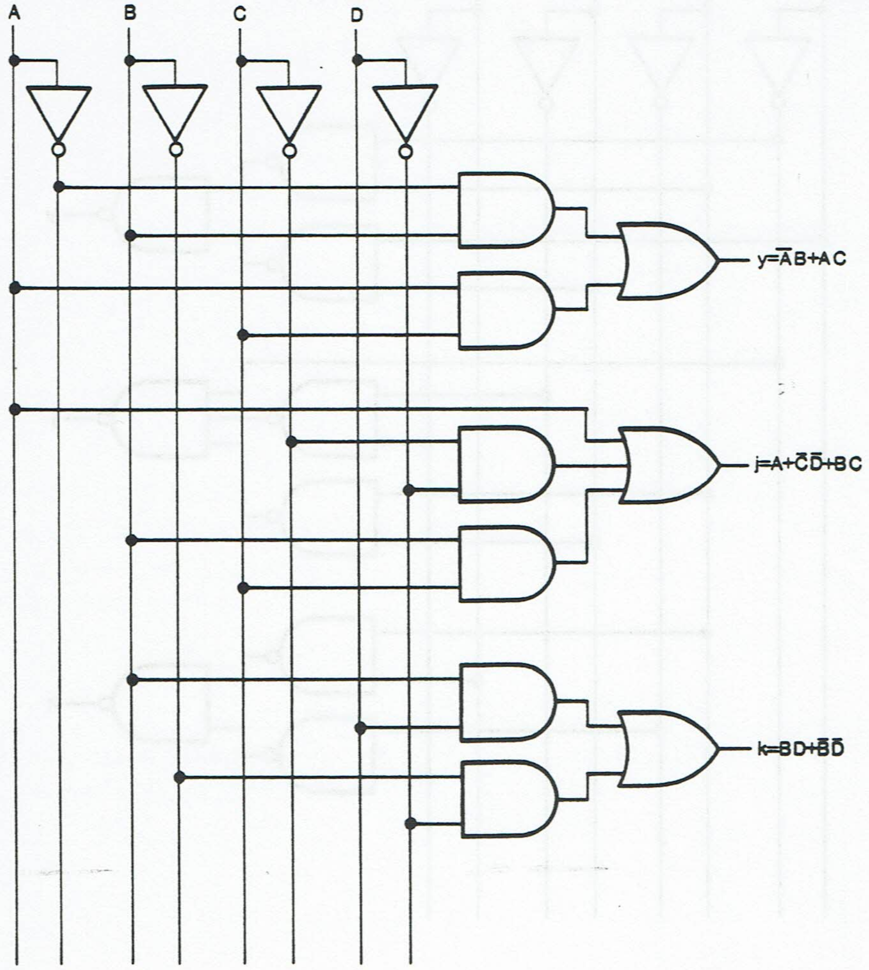

A
B
C
D
$y=\overline{A}B+AC$
$j=A+\overline{C}\overline{D}+BC$
$k=BD+\overline{B}\overline{D}$

Figure 2-28

AND/OR designs for question 26.

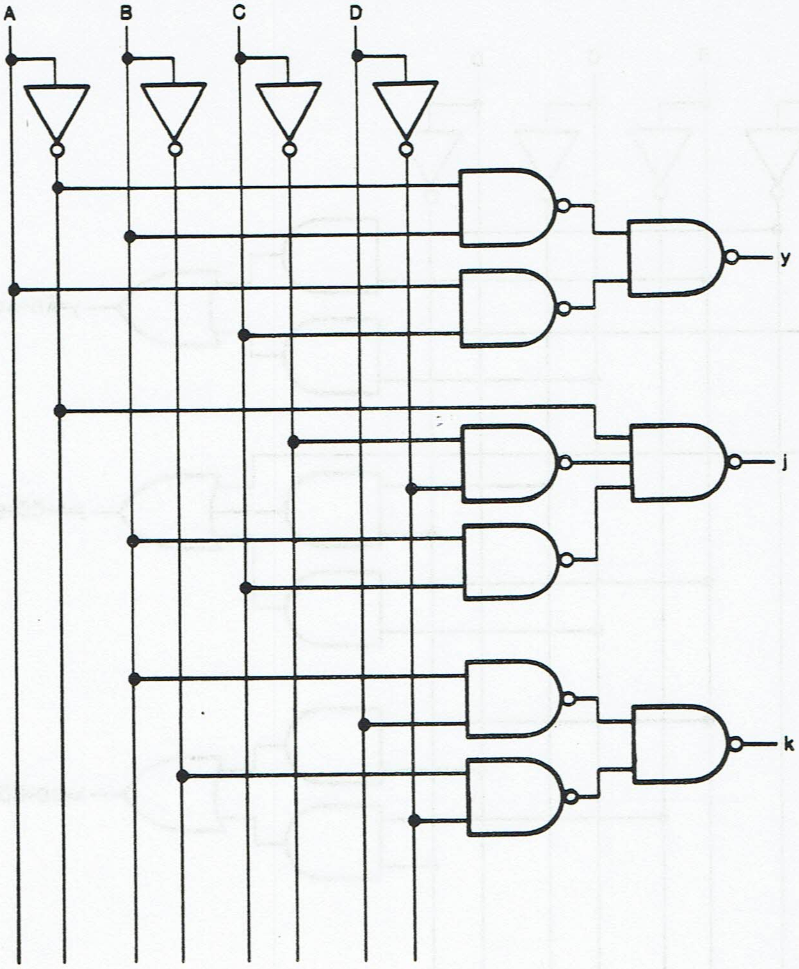


Figure 2-29

NAND designs for question 28.

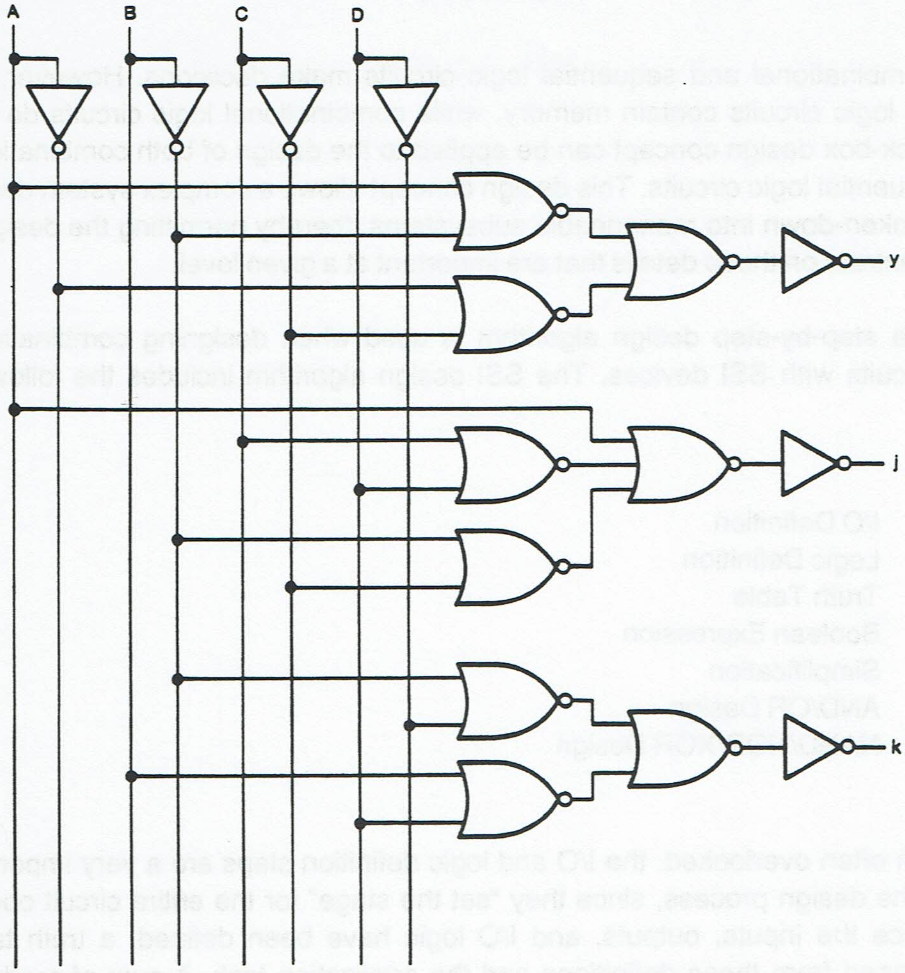


Figure 2-30

NOR designs for question 29.

SUMMARY

Both combinational and sequential logic circuits make decisions. However, sequential logic circuits contain memory, while combinational logic circuits do not. The black-box design concept can be applied to the design of both combinational and sequential logic circuits. This design concept allows a complex system design to be broken-down into manageable subsystems, thereby permitting the designer to concentrate on those details that are important at a given level.

A simple step-by-step design algorithm is used when designing combinational logic circuits with SSI devices. The SSI design algorithm includes the following steps:

- I/O Definition
- Logic Definition
- Truth Table
- Boolean Expression
- Simplification
- AND/OR Design
- NAND/NOR/XOR Design

Although often overlooked, the I/O and logic definition steps are a very important part of the design process, since they "set the stage" for the entire circuit operation. Once the inputs, outputs, and I/O logic have been defined, a truth table is developed from these definitions and the application task. A sum-of-products Boolean expression is then written directly from the truth table.

The original Boolean expression must be simplified, since a simplified expression results directly in a simplified circuit. The laws of Boolean algebra can be applied to simplify some Boolean expressions. However, more complex Boolean expressions require the use of a graphical technique called Karnaugh mapping. Karnaugh mapping simplifies the simplification process.

An AND/OR circuit can be derived directly from the simplified Boolean expression. However, this design is usually converted to a NAND/NOR/XOR circuit since these devices are more economical, readily available, and easier to implement than AND/OR devices. In fact, the most common digital circuit component is the NAND gate. Any combinational logic circuit can be constructed using only NAND gates.

Finally, you must decide which family and series of digital ICs to use. The application will usually dictate this decision according to the advantages/disadvantages and features of each family. Then, once a given family and series of ICs is chosen, you must decide which particular ICs will result in a minimum circuit chip count. This will guarantee the most economical design.

Before you take the Unit Examination, perform Experiments 6 and 7 in Unit 8.

UNIT EXAMINATION

1. Feedback signals are found in:
   A. Combinational logic circuits.
   B. Logic gates.
   C. Sequential logic circuits.
   D. Memory elements.

2. A decoder is an example of a:
   A. Combinational logic circuit.
   B. Logic gate.
   C. Sequential logic circuit.
   D. Memory element.

3. A circuit that converts binary to some other code is a(n):
   A. Encoder.
   B. Memory element.
   C. Multiplexer.
   D. Decoder.

4. A 7-segment LED whose elements are activated by a logic 1 level is called a:
   A. Common anode LED.
   B. Common emitter LED.
   C. Common cathode LED.
   D. Common collector LED.

5. The SOP expression for the following truth table is:

A	B	C	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

A. $y = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}C + AB\overline{C}$

B. $y = \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}\overline{C} + AB\overline{C}$

C. $y = \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}\overline{C} + ABC$

D. $y = \overline{A}\overline{B}C + \overline{A}BC + ABC$

6. The numerical representation of the following expression is:

$$\overline{x}y\overline{z} + xy\overline{z} + xyz + \overline{x}\overline{y}\overline{z}$$

A. f(A,B,C) = Σ(0,2,6,7)
B. f(x,y,z) = Σ(0,1,5,7)
C. f(A,B,C) = Σ(0,1,5,7)
D. f(x,y,z) = Σ(0,2,6,7)

7. The expression $AB + \overline{A}\overline{B}$ is equivalent to:
A. $A \oplus B$
B. $\overline{AB}$
C. AB
D. $\overline{A \oplus B}$

8. The Boolean equation, $y = \overline{A}B + AB + A\overline{B}$, reduces to:
A. $A \oplus B$
B. $A + B$
C. AB
D. $(A \oplus B) + A$

9. The K-map in Figure 2-15c generates the expression:
A. $\overline{A} + \overline{C}\overline{B}$
B. $C + B\overline{D}$
C. $\overline{C} + \overline{B}D$
D. $A + CB$

10. The K-map in Figure 2-15e generates the expression:
A. $BD + \overline{B}\overline{D}$
B. $B\overline{D} + \overline{B}D$
C. $BD + BD$
D. None of these

11. Given the following truth table:

A	B	C	D	m	n
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	1	1
1	0	1	0	X	X
1	0	1	1	X	X
1	1	0	0	X	X
1	1	0	1	X	X
1	1	1	0	X	X
1	1	1	1	X	X

The Boolean expression for m reduces to:

A. $\bar{A}BC + A\bar{B}\bar{C} + \bar{A}B\bar{C}\bar{D}$
B. $\bar{A} + BC + B\bar{D}$
C. $A + BC + B\bar{D}$
D. $A + \bar{B}C + B\bar{C}D$

12. The Boolean expression for n in question 11 reduces to:
A. $A + \bar{B}C + B\bar{C}D$
B. $A + BC + B\bar{D}$
C. $A + \bar{A}\bar{B}C + \bar{A}B\bar{C}D$
D. $\bar{A} + \bar{B}C + B\bar{C}D$

13. The simplest Boolean equation for the "f" segment of a common cathode LED using a BCD input is:
A. $f = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D$
B. $f = A + \bar{C}\bar{D} + B\bar{C} + B\bar{D}$
C. $f = A + \bar{C}\bar{D} + B\bar{C} + BC\bar{D}$
D. $f = CD + \bar{A}\bar{B}D + \bar{A}\bar{B}C$

14. How many and what size AND/OR gates would it take to implement the "f" segment equation in question 13?
A. Six 4-bit AND gates and a 6-bit OR gate.
B. Three 2-bit AND gates and a 4-bit OR gate.
C. Two 2-bit AND gates, a 3-bit AND gate, and a 4-bit OR gate.
D. One 2-bit AND gate, two 3-bit AND gates, and a 3-bit OR gate.

15. Refer to the circuits in Figure 2-31. Which circuit is designed to implement the equation, $y = \overline{A}B + \overline{B}\,\overline{C} + \overline{C}\,\overline{D}$?
    - A. Circuit a.
    - B. Circuit b.
    - C. Circuit c.
    - D. Circuit d.

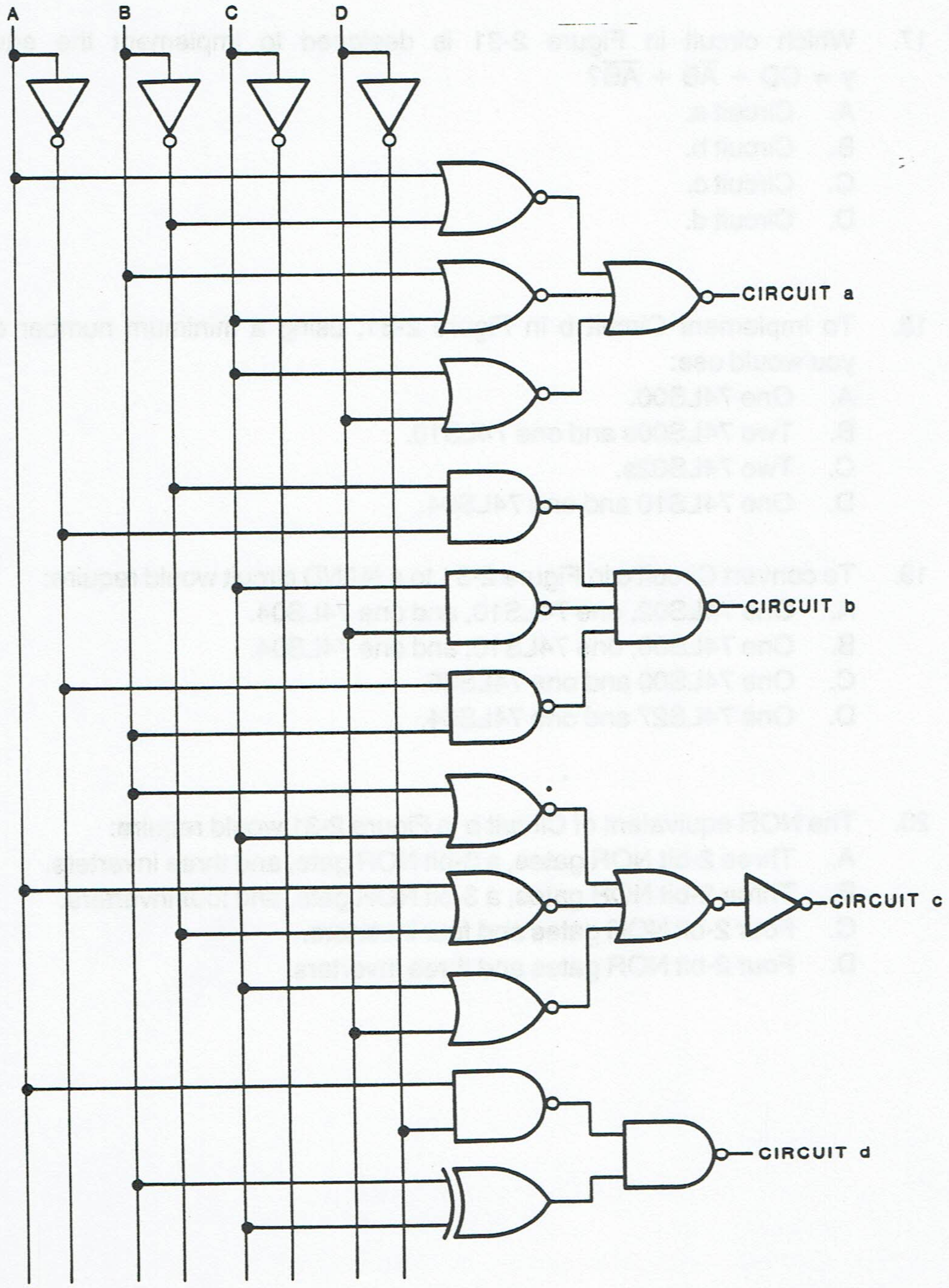


Figure 2-31

Circuits for questions 15 through 20.

16. Which circuit in Figure 2-31 is designed to implement the equation, $y = A\overline{D} + BC + \overline{B}\overline{C}$?
    A. Circuit a.
    B. Circuit b.
    C. Circuit c.
    D. Circuit d.

17. Which circuit in Figure 2-31 is designed to implement the equation, $y = CD + \overline{A}B + \overline{A}\overline{B}$?
    A. Circuit a.
    B. Circuit b.
    C. Circuit c.
    D. Circuit d.

18. To implement Circuit b in Figure 2-31, using a minimum number of ICs, you would use:
    A. One 74LS00.
    B. Two 74LS00s and one 74LS10.
    C. Two 74LS02s.
    D. One 74LS10 and one 74LS04.

19. To convert Circuit c in Figure 2-31 to a NAND circuit would require:
    A. One 74LS02, one 74LS10, and one 74LS04.
    B. One 74LS00, one 74LS10, and one 74LS04.
    C. One 74LS00 and one 74LS86.
    D. One 74LS27 and one 74LS04.

20. The NOR equivalent of Circuit b in Figure 2-31 would require:
    A. Three 2-bit NOR gates, a 3-bit NOR gate, and three inverters.
    B. Three 2-bit NOR gates, a 3-bit NOR gate, and four inverters.
    C. Four 2-bit NOR gates and four inverters.
    D. Four 2-bit NOR gates and three inverters.

EXAMINATION ANSWERS

1. C — Feedback signals are found in sequential logic circuits.

2. A — A decoder is an example of a combinational logic circuit.

3. D — A circuit that converts binary to some other code is a decoder.

4. C — A 7-segment LED whose elements are activated by a logic 1 is called a common cathode LED.

5. C — The SOP expression for the truth table shown in question 5 is $y = \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}\overline{C} + ABC$.

6. D — The numerical representation of the expression $\overline{x}y\overline{z} + xy\overline{z} + xyz + \overline{x}\overline{y}\overline{z}$ is $f(x,y,z) = \Sigma(0,2,6,7)$.

7. D — The expression $AB + \overline{A}\overline{B}$ is equivalent to $\overline{A \oplus B}$.

8. B — The Boolean equation, $y = \overline{A}B + AB + A\overline{B}$, reduces to $A + B$.

9. B — The K-map in Figure 2-15c generates the expression $C + B\overline{D}$.

10. A — The K-map in Figure 2-15e generates the expression $BD + \overline{B}\overline{D}$.

11. C — Given the truth table in question 11, the Boolean expression for m reduces to $A + BC + B\overline{D}$.

12. A — The Boolean expression for n in question 11 reduces to $A + \overline{B}C + B\overline{C}D$.

13. B — The simplest Boolean equation for the "f" segment of a common cathode LED using a BCD input is:

$$f = A + \overline{C}\overline{D} + B\overline{C} + B\overline{D}.$$

14. B — The number and size of AND/OR gates it would take to implement the "f" segment equation in question 13 is three 2-bit AND gates and one 4-bit OR gate.

15. C — Referring to the circuits in Figure 2-31, the circuit designed to implement the equation $y = \overline{A}B + \overline{B}\overline{C} + \overline{C}\overline{D}$ is circuit c.

16. D — The circuit in Figure 2-31 that is designed to implement the equation $y = A\overline{D} + BC + \overline{BC}$ is circuit d.

17. B — The circuit in Figure 2-31 that is designed to implement the equation $y = CD + \overline{A}B + \overline{AB}$ is circuit b.

18. B — To implement Circuit b in Figure 2-31, using a minimum number of ICs, you would use two 74LS00s and one 74LS10.

19. B — To convert Circuit c in Figure 2-31 to a NAND circuit will require one 74LS00, one 74LS10, and one 74LS04.

20. B — The NOR equivalent of circuit b in Figure 2-31 requires three 2-bit NOR gates, a 3-bit NOR gate, and four inverters.

UNIT 3

COMBINATIONAL LOGIC DESIGN USING MSI AND LSI DEVICES

CONTENTS

Introduction . . . . Page 3-3

Unit Objectives . . . . Page 3-4

Unit Activity Guide . . . . Page 3-5

Multiplexers . . . . Page 3-6

Decoders . . . . Page 3-41

Read Only Memories (ROMs) . . . . Page 3-65

Programmable Logic Arrays (PLAs) . . . . Page 3-83

Summary . . . . Page 3-89

Unit Examination . . . . Page 3-90

Examination Answers . . . . Page 3-94

INTRODUCTION

As a natural extension to Unit 2, this unit will show you how to incorporate higher levels of integration into your design. There are many combinational logic functions that have been standardized within MSI and LSI devices. For example, in the last Unit, several examples and problems involved the design of logic circuits to illuminate the segments of a seven-segment LED. As it turns out, these designs make good learning tools. However, they would never be produced on a commercial basis, since there already is a single MSI device that performs the entire decoding function. In practice, if you knew about such a device, you would never waste a lot of time and effort attempting to design a seven-segment LED decoder from logic equations using SSI devices. For this reason, it is important that you know what standard digital functions have been integrated into MSI and LSI devices.

Unfortunately, not all combinational logic operations have been standardized into MSI and LSI devices. To do so would require a device for each specific logic design. From a manufacturing point of view this is not practical or possible. As a result, you will still have to write logic equations and use SSI devices in much of your design work. However, during the black-box design concept stage, you must consider what MSI and/or LSI devices are available to implement various portions of the design. These then become black-boxes by themselves and any remaining logic is implemented using SSI devices. Logic design with MSI and LSI devices must occur at the system black-box level, since once you begin to write logic equations it is almost impossible to "see the forest for the trees."

In this Unit, you will learn about the two most common MSI devices that are used to implement combinational logic: the multiplexer and the decoder. You will be acquainted with several commercial IC devices in order to give you a "feel" for what standard MSI devices are available. In addition, you will learn how to simplify and implement logic equations using MSI multiplexers and decoders. Finally, the discussion of MSI devices will be followed by a discussion of LSI devices that are available for use in combinational logic design. The emphasis here will be on read only memories, or ROMs, and programmable logic arrays, or PLAs. Now, let's get down to business.

UNIT OBJECTIVES

When you complete this unit, you should be able to:

1. Explain the operation of the following MSI devices:
   - A. Multiplexer
   - B. Decoder
   - C. Demultiplexer
2. Design multiplexer circuits of any size using smaller multiplexers.
3. Describe several multiplexer IC devices.
4. Design combinational logic circuits using MSI multiplexers to implement the design.
5. Compare different multiplexer design techniques.
6. Determine how many commercial ICs are required to implement a given design using different multiplexer design techniques.
7. Describe several decoder IC devices.
8. Design combinational logic circuits using MSI decoders to implement the design.
9. Explain how decoders are used for address decoding in computer systems.
10. Explain how to use a decoder as a demultiplexer.
11. Describe the operation of ROMs and PLAs.
12. Discuss the advantages and disadvantages of using different types of ROM devices to implement a design.
13. Design combinational logic circuits using LSI ROMs and PLAs to implement the design.
14. State the reasons for using a PLA over a ROM for implementing complex logic functions.

UNIT ACTIVITY GUIDE

- ☐ Read section on "Multiplexers."
- ☐ Answer Self-Test Review Questions 1-12.
- ☐ Read section on "Decoders."
- ☐ Answer Self-Test Review Questions 13-21.
- ☐ Perform Experiments 8 and 9 in Unit 8.
- ☐ Read section on "Read Only Memories (ROMs)."
- ☐ Answer Self-Test Review Questions 22-30.
- ☐ Read section on "Programmable Logic Arrays (PLAs)."
- ☐ Answer Self-Test Review Questions 31-35.
- ☐ Study Summary.
- ☐ Perform Experiment 10 in Unit 8.
- ☐ Complete Unit Examination.
- ☐ Check Examination Answers.

MULTIPLEXERS

The multiplexer is one of the most useful MSI devices available. In addition to performing a digital switching function, it can also be used to implement logic equations.

Multiplexer Types

Functionally, a multiplexer is nothing more than an electronic switch as shown in Figure 3-1. It consists of several digital input channels labeled D0, D1, D2, and so on, and a single output channel labeled y. The idea is simple: the switch selects which one of the inputs is passed on to the output. For this reason, a multiplexer is sometimes called a **data selector**.

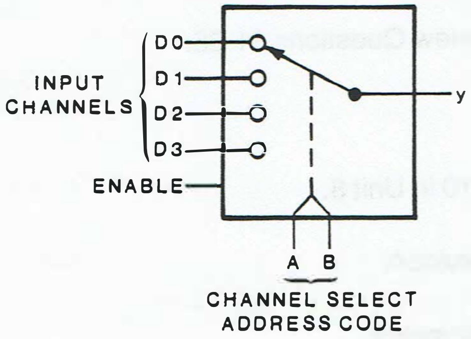


Figure 3-1

A multiplexer is simply an electronic switch used for data selection.

When the switch is in the position shown in Figure 3-1, the output, y, equals input D0. In symbols, y = D0. If the switch were moved down to the next input line, the output is connected to input D1, or y = D1. In the third switch position y = D2, and in the last position y = D3.

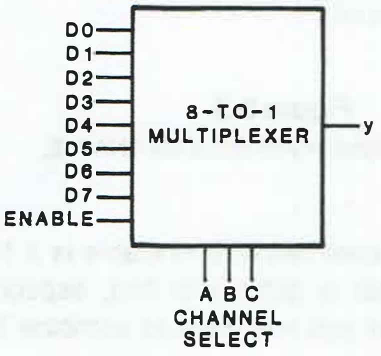
Control of the electronic switch position is provided by the channel select input lines, A and B. If the logic on AB is 00, channel D0 is selected. Likewise, if the logic on AB is 01, channel D1 is selected. With this idea, the multiplexer shown in Figure 3-1 has the following truth table:

A	B	y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

The channel select logic is sometimes called an **address code**, since you can think of the multiplexer input lines as having addresses, 0 through n, where n is the address of the last input line. Notice that the address of the last input line will always be equal to the number of input lines, minus one. The reason for this is that digital addresses, and labeling in general, always start with 0, not 1, as you might be accustomed to seeing.

You can think of the ENABLE line in Figure 3-1 as an on/off switch. When the ENABLE line is active, the multiplexer is on and operates per the truth table shown earlier. If the ENABLE line is inactive, the multiplexer is off and its output is held at a constant logic level, 0 or 1, depending on the specific multiplexer used. Of course, ENABLE can be active low or active high. In Figure 3-1, ENABLE is active high. If ENABLE were active low, the ENABLE line would be bubbled where it connects to the multiplexer logic symbol.

The multiplexer in Figure 3-1 is called a 4-to-1 multiplexer since it can select one of four inputs to appear on the output. An 8-to-1 multiplexer is shown in Figure 3-2 along with its associated truth table. Here, the idea is the same: one of eight inputs, D0 through D7, is selected by the channel select lines to appear on output line y.



A	B	C	y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

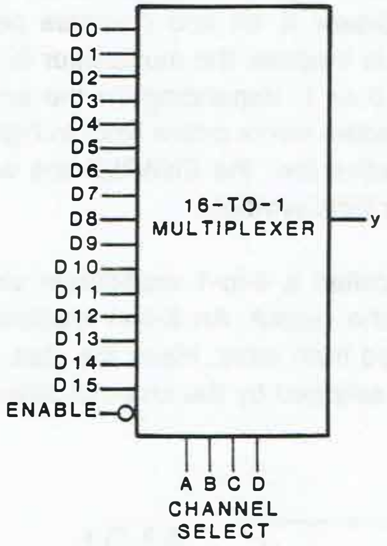
Figure 3-2
An 8-to-1 multiplexer.

Example 3-1

How many channel select lines are required for a 16- to-1 multiplexer. Sketch the multiplexer and determine its truth table. Assume an active low ENABLE line.

To select one of sixteen inputs, there must be at least four channel select lines. To determine this, you must ask yourself: "Self, what power of 2 will give me 16?" The answer is, "2 raised to the **fourth** power gives you 16." Thus, four channel select lines are required.

A 16-to-1 multiplexer is shown in Figure 3-3, along with its associated truth table. Notice that the ENABLE line is bubbled, indicating an active low condition.



A	B	C	D	y
0	0	0	0	D0
0	0	0	1	D1
0	0	1	0	D2
0	0	1	1	D3
0	1	0	0	D4
0	1	0	1	D5
0	1	1	0	D6
0	1	1	1	D7
1	0	0	0	D8
1	0	0	1	D9
1	0	1	0	D10
1	0	1	1	D11
1	1	0	0	D12
1	1	0	1	D13
1	1	1	0	D14
1	1	1	1	D15

Figure 3-3

A 16-to-1 multiplexer with active low ENABLE.

Currently, the largest multiplexer commercially available is a 16-to-1 multiplexer. In fact, even a 16-to-1 multiplexer is difficult to find, especially in CMOS. As a result, there will be times when you will have to combine two or more small multiplexers to design a large one.

The process of designing larger multiplexers from smaller ones is relatively simple as illustrated by Figure 3-4. Here, four 8-to-1 multiplexers and a 4-to-1 multiplexer have been combined to form a 32-to-1 multiplexer. The address code for the multiplexer, ABCDE, must be a 5-bit code since 2 is raised to the fifth power to get 32. The three least significant address code bits, CDE, are used to select one of eight inputs for each 8-to-1 multiplexer. The logic of the selected input line for each of the 8-to-1 multiplexers is applied to the four inputs of the 4-to-1 multiplexer. There, the two most significant address code bits, AB, select one of the four inputs to be passed to the output of the 4-to-1 multiplexer.

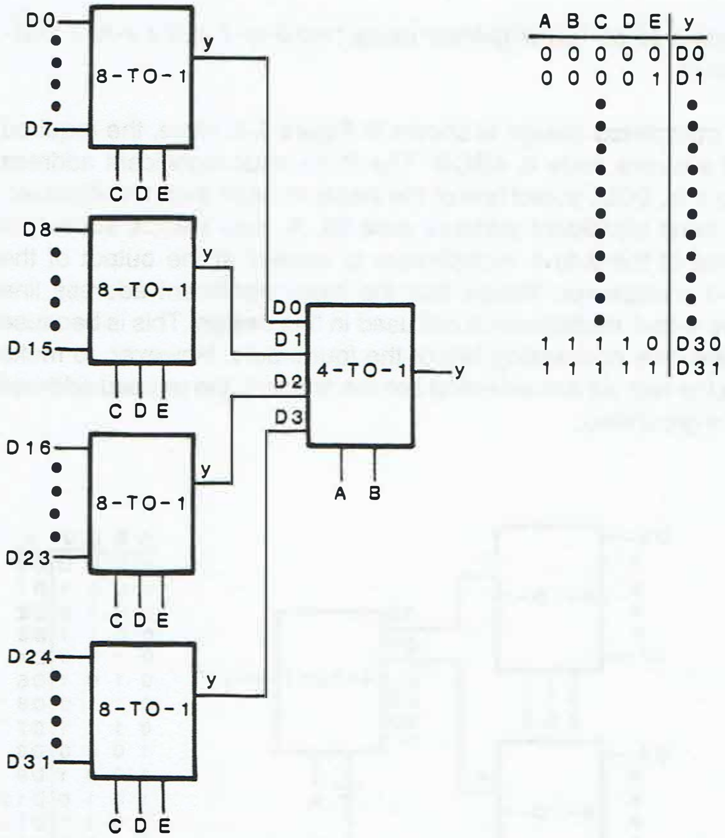


Figure 3-4

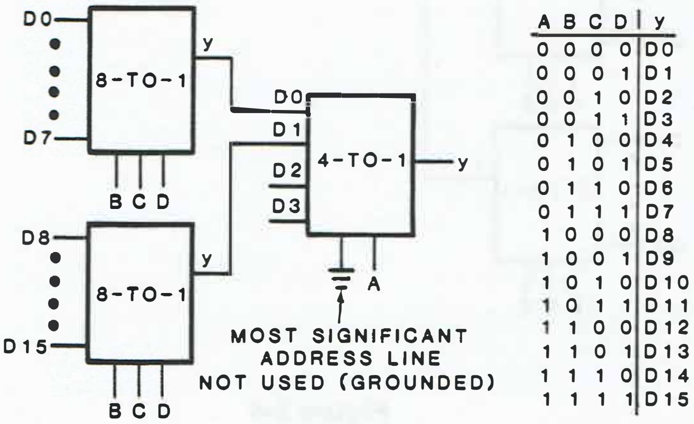
A 32-to-1 multiplexer constructed from four 8-to-1 and one 4-to-1 multiplexers.

For instance, an address code of ABCDE = 01011 will select input line D11. First, address code bits CDE = 011 selects the fourth input line of each 8-to-1 multiplexer. Then, address code bits AB = 01 selects the output of the second 8-to-1 multiplexer. Notice that the fourth input line to the second 8-to-1 multiplexer is D11. Because of the multiplexer decoding scheme, you can also determine the selected input line by simply converting the binary address code to decimal. Binary 01011 equals decimal 11.

Example 3-2

Design a 16-to-1 multiplexer using two 8-to-1 and a 4-to-1 multiplexer.

The completed design is shown in Figure 3-5. Here, the required 4-bit address code is ABCD. The three least significant address code bits, BCD, select one of the inputs to each 8-to-1 multiplexer. The most significant address code bit, A, then selects the output of one of the 8-to-1 multiplexers to appear at the output of the 4-to-1 multiplexer. Notice that the most significant address line of the 4-to-1 multiplexer is not used in this design. This is because we are only addressing two of the four inputs. However, to make sure the two we are selecting are the first two, the unused address line is grounded.

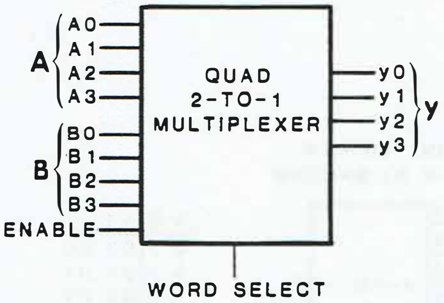


A	B	C	D	y
0	0	0	0	D0
0	0	0	1	D1
0	0	1	0	D2
0	0	1	1	D3
0	1	0	0	D4
0	1	0	1	D5
0	1	1	0	D6
0	1	1	1	D7
1	0	0	0	D8
1	0	0	1	D9
1	0	1	0	D10
1	0	1	1	D11
1	1	0	0	D12
1	1	0	1	D13
1	1	1	0	D14
1	1	1	1	D15

Figure 3-5

A 16-to-1 multiplexer constructed from two 8-to-1 and one 4-to-1 multiplexer.

Another type of multiplexer is shown in Figure 3-6. This multiplexer could be called a **word selector**, since it is used to select one of two 4-bit digital words: **A** or **B**. Notice that input **A** = A3 A2 A1 A0 and input **B** = B3 B2 B1 B0, while output **y** = y3 y2 y1 y0. From now on, when you see a single **bold face** letter it will designate a multi-bit input or output word.



WORD SELECT	y
0	A
1	B

Figure 3-6

A quad 2-to-1 multiplexer is a 4-bit word selector.

The multiplexer in Figure 3-6 is called a **quad 2-to-1** multiplexer since it selects one of two input words, each with four (quad) bits. Only one select line is required since there are only two things to select, word **A** or word **B**. The associated truth table shows that word **A** is selected when the word select line is low, while word **B** is selected when the word select line is high.

Multiplexer IC Devices

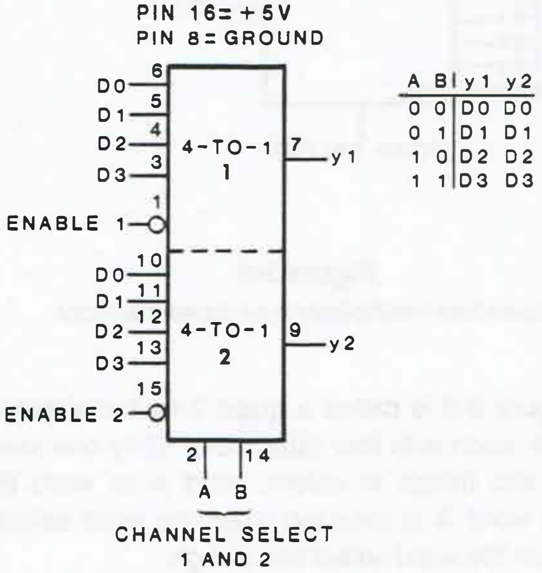
There are several multiplexer devices that are standard throughout the industry. Table 3-1 provides a representative listing of these devices.

Table 3-1: A Representative Listing of Multiplexer Devices

Function	TTL	CMOS
Dual 4-to-1	74153	74HC153 or 4539/14539
8-to-1	74151	74C151 or 74HC151 or 4512/14512
16-to-1	74150	None
Quad 2-to-1	74157/74158	74C157/74C158 or 74HC157/74HC158

THE 74153 DUAL 4-to-1 MULTIPLEXER

A functional I/O diagram and truth table of this device is provided in Figure 3-7. First, notice that there are two 4-to-1 multiplexers in the same IC package. This is the reason it is called a "dual" 4-to-1 multiplexer. Each multiplexer section has four separate input lines, an output line, and an enable line. The two channel select lines are common to both multiplexer sections. Consequently, selecting any given channel for one multiplexer also selects the same channel for the other multiplexer.



A	B	y1	y2
0	0	D0	D0
0	1	D1	D1
1	0	D2	D2
1	1	D3	D3

Figure 3-7

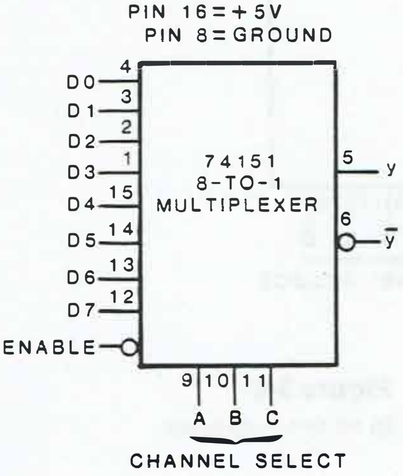
The 74153 is a dual 4-to-1 multiplexer.

There are two separate ENABLE lines in Figure 3-7, one for each multiplexer section. These lines are active low, as indicated by their bubbled inputs. When a given multiplexer section is disabled (ENABLE high) the output of that section is held at a constant logic low level, regardless of its input or channel select logic.

Before we go any further, I should mention that in some texts and manufacturer's literature, you will see the order of the input lines reversed. That is, instead of ABC you will see CBA, where the variable A is the least significant bit. This usually leads to confusion since many inferior texts will use the ABC order when dealing with SSI logic, then switch to the CBA order when explaining MSI logic. For consistency within this text, I will always use the ABC order where the variable A is always the most significant bit in any multi-bit word. The important thing is to keep track of your ABC's when dealing with digital logic.

THE 74151 8-to-1 MULTIPLEXER

This device is shown in Figure 3-8. Notice that it has eight input lines, an active low enable line, and three channel select lines. In addition, there are two output lines, y and $\bar{y}$. From the truth table in Figure 3-8 you can see that $\bar{y}$ is simply the complement of y.



A	B	C	y	$\bar{y}$
0	0	0	D0	$\overline{D0}$
0	0	1	D1	$\overline{D1}$
0	1	0	D2	$\overline{D2}$
0	1	1	D3	$\overline{D3}$
1	0	0	D4	$\overline{D4}$
1	0	1	D5	$\overline{D5}$
1	1	0	D6	$\overline{D6}$
1	1	1	D7	$\overline{D7}$

Figure 3-8
The 74151 8-to-1 multiplexer.

THE 74150 16-to-1 MULTIPLEXER

There are really no big surprises in this device as you can see from Figure 3-9. It has sixteen input lines, a single output line, four channel select lines, and an active low ENABLE line. The output line is held at a constant high level when it is disabled (ENABLE high).

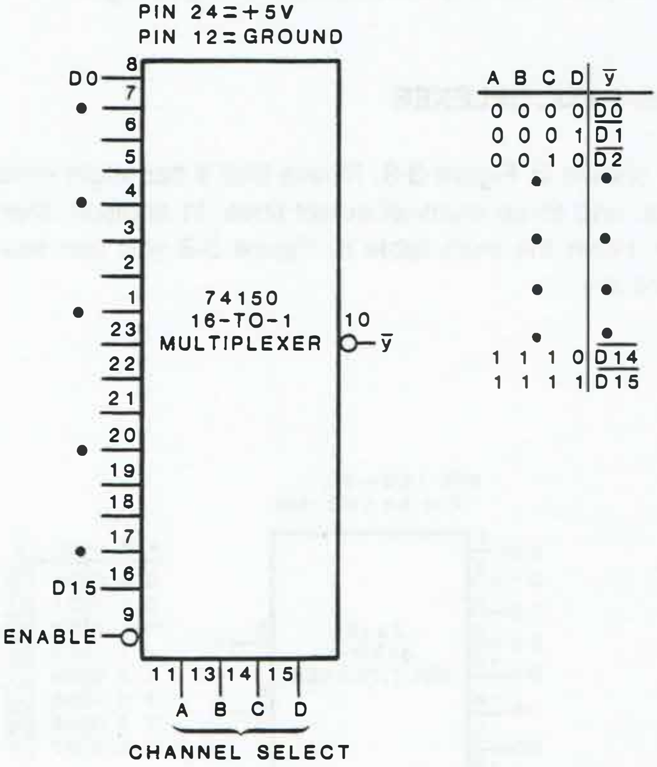


Figure 3-9

The 74150 16-to-1 multiplexer.

The only peculiarity of this device is that its output produces the complement of the selected input. This feature is designated by the bubbled output line. I should also mention that this device is only available in a standard TTL package. Other TTL or CMOS versions of this device are not readily available.

THE 74157/74158 QUAD 2-to-1 MULTIPLEXER

This device is shown in Figure 3-10. A single channel select line selects between one of two 4-bit input words, **A** or **B**. The selected word is transferred to the four output lines when the ENABLE line is active low. When the ENABLE line is inactive, or high, all the outputs are high, regardless of the input logic. The 74158 is an inverted output version of the 74157.

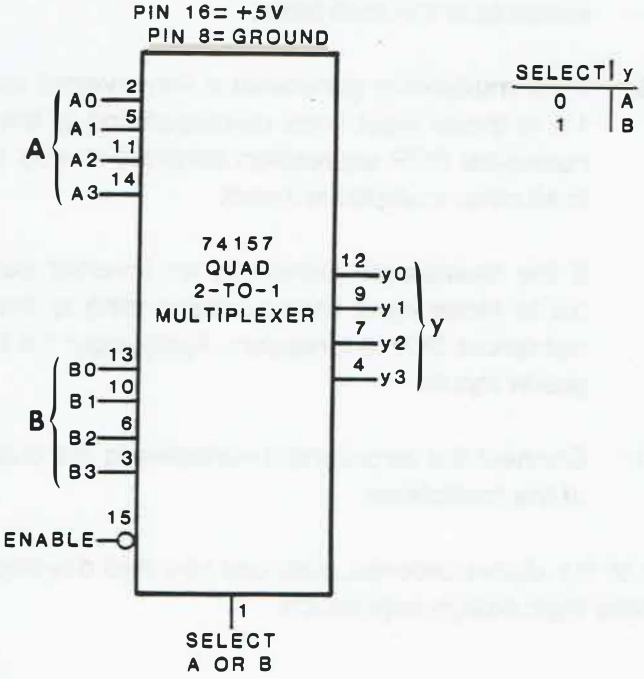


Figure 3-10
The 74157 quad 2-to-1 multiplexer.

Implementing Logic Equations With Multiplexers

Multiplexers are commonly used to implement logic equations since they can reduce the overall chip-count in a circuit. This can decrease the overall cost of the circuit. There are basically three solutions to implementing logic equations with multiplexers. The first I call the "obvious solution" since, as you will soon see, the idea is rather obvious. The second and third methods I call the n-1 and n-2 methods, respectively. The reason for these designations will become clear after you see how they work.

THE OBVIOUS SOLUTION

The obvious solution involves the direct implementation of a truth table by a multiplexer. The following steps summarize the design procedure:

Step 1: Write an SOP expression from the truth table, in **numerical form.**

Step 2: Select a 2^n-to-1 multiplexer, where n is the number of input variables in the truth table.

Step 3: If the multiplexer generates a non-inverted output, apply logic 1's to those input lines corresponding to the numbers in the numerical SOP expression obtained in step 1. Apply logic 0's to all other multiplexer inputs.

If the multiplexer generates an inverted output, apply logic 0's to those input lines corresponding to the numbers in the numerical SOP expression. Apply logic 1's to all other multiplexer inputs.

Step 4: Connect the circuit input variables to the channel select lines of the multiplexer.

As an example of the above process, suppose you had developed the following truth table for some logic design application:

INPUT	A	B	C	D	y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

Step 1 of the process requires you to write an SOP expression for the above table in numerical form. A logic 1 output appears for inputs of 1, 2, 5, 6, 8, 10, 12, and 13 in the above table. Consequently, the SOP expression is:

$$y = \Sigma\,(1,2,5,6,8,10,12,13)$$

In step 2, you must pick a multiplexer. There are four input variables in the truth table, and $2^4 = 16$. Thus, a 16-to-1 multiplexer will do the job. Let's use a 74150.

Next, you simply apply logic to the 74150 input lines as required by step 3. Recall that the 74150 generates an inverted output. As a result, logic 0's must be applied to the lines listed in the SOP expression, while logic 1's are applied to all other input lines. The proper logic application is shown in Figure 3-11.

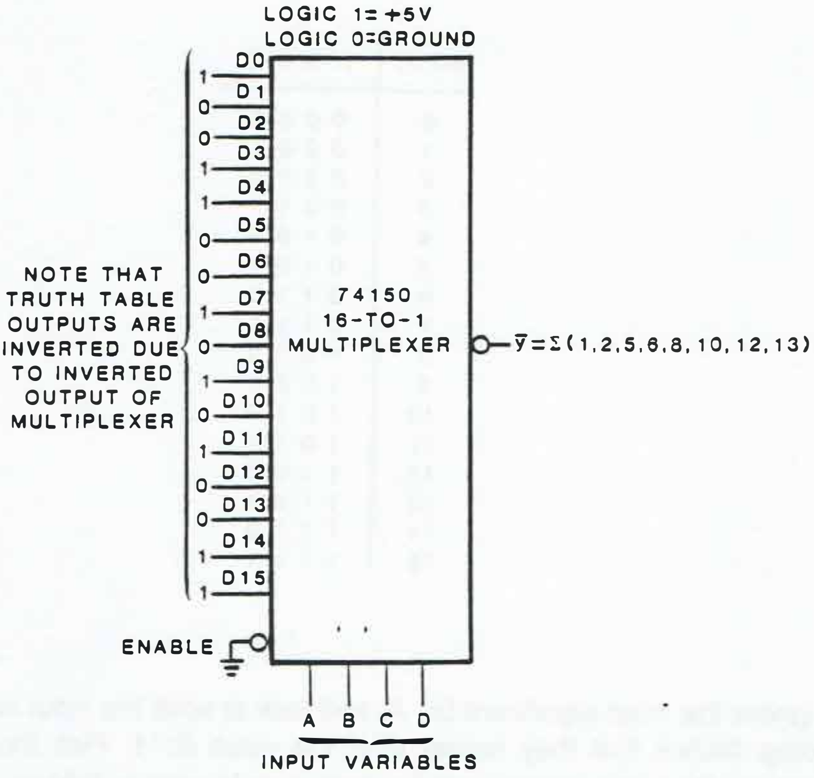


Figure 3-11

The obvious solution applies the truth table outputs directly to the multiplexer inputs.

Finally, step 4 requires you to connect the circuit inputs to the address lines of the multiplexer as shown in the figure.

To check our design, suppose an address of ABCD 1010 is applied to the multiplexer. This address selects input line D10 which is tied low. The inverted output from the multiplexer is, therefore, a logic 1. Notice that a logic 1 output appears in the truth table opposite an input value of 1010. You might want to test some other cases to assure a valid design.

THE n-1 SOLUTION

You can decrease the size of the multiplexer required to implement a logic equation by doing a little extra thinking during the design process. Look at the following truth table listing for four input variables:

INPUT	A B C D
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

Now, ignore the most significant bit, A, and look at what the input variables BCD are doing. Notice that they repeat after the value 0111. Pick those values of BCD that are the same and you can list them in eight pairs as follows:

	INPUT	B C D
Pair 0	0	0 0 0
	8	0 0 0
Pair 1	1	0 0 1
	9	0 0 1
Pair 2	2	0 1 0
	10	0 1 0
Pair 3	3	0 1 1
	11	0 1 1
Pair 4	4	1 0 0
	12	1 0 0
Pair 5	5	1 0 1
	13	1 0 1
Pair 6	6	1 1 0
	14	1 1 0
Pair 7	7	1 1 1
	15	1 1 1

The next step is to write the truth table outputs next to the inputs in the above pair listing. Let's use the same truth table that we used for the obvious solution example above. The resulting tabulation is:

	INPUT	B C D	y
Pair 0	0	0 0 0	0
	8	0 0 0	1
Pair 1	1	0 0 1	1
	9	0 0 1	0
Pair 2	2	0 1 0	1
	10	0 1 0	1
Pair 3	3	0 1 1	0
	11	0 1 1	0
Pair 4	4	1 0 0	0
	12	1 0 0	1
Pair 5	5	1 0 1	1
	13	1 0 1	1
Pair 6	6	1 1 0	1
	14	1 1 0	0
Pair 7	7	1 1 1	0
	15	1 1 1	0

From this table, you see that there are only four possible logic combinations of the output, y, for any given input pair. They are: 00, 01, 10, and 11. Let's represent these logic pairs as follows:

y = 00, pair value = 0
y = 01, pair value = A
y = 10, pair value = $\bar{A}$
y = 11, pair value = 1

Pair value A is the most significant bit of the address code, while $\bar{A}$ is its complement. Assigning the **pair values** to each number pair gives you:

	INPUT	B C D	y	Pair Value
Pair 0	0	0 0 0	0	A
	8	0 0 0	1	
Pair 1	1	0 0 1	1	$\bar{A}$
	9	0 0 1	0	
Pair 2	2	0 1 0	1	1
	10	0 1 0	1	
Pair 3	3	0 1 1	0	0
	11	0 1 1	0	
Pair 4	4	1 0 0	0	A
	12	1 0 0	1	
Pair 5	5	1 0 1	1	1
	13	1 0 1	1	
Pair 6	6	1 1 0	1	$\bar{A}$
	14	1 1 0	0	
Pair 7	7	1 1 1	0	0
	15	1 1 1	0	

The last step is to equate the pair numbers with the input lines of a multiplexer, and apply the pair value to the respective multiplexer input lines. Since there are eight pairs, an 8-to-1 multiplexer will do the job. The final design, using a 74151 multiplexer, is shown in Figure 3-12.

Using the above results, the value obtained for Pair 0 is A. Thus, the input variable A is connected to input line 0 of the multiplexer. The pair value for Pair 1 is $\bar{A}$. As a result, $\bar{A}$ is connected to input line 1. Input line 2 of the multiplexer is tied high, since the value of pair 2 is 1. Likewise, input line 3 is tied low, since the value of pair 3 is 0. The remaining connections are made in the same manner. Finally, the three least significant input variables, BCD, are connected to the channel select lines of the multiplexer as shown.

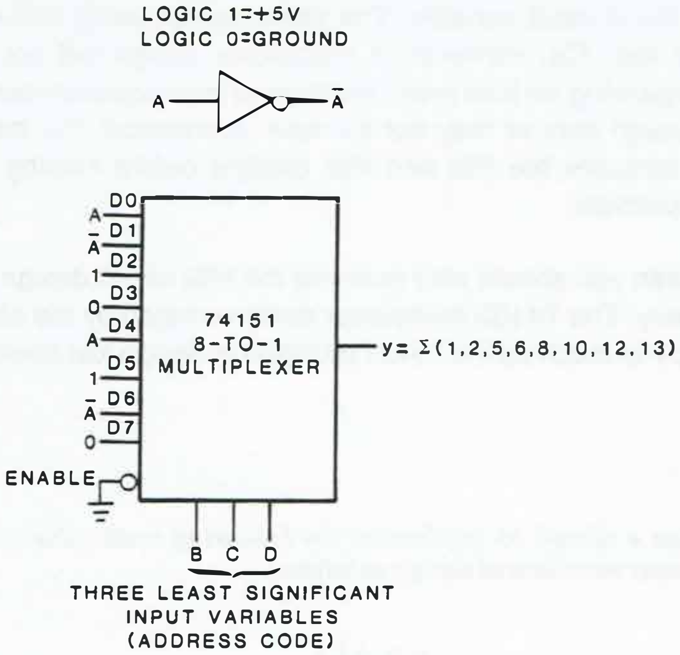


Figure 3-12

The n-1 solution reduces the size of the multiplexer by one-half over the obvious solution.

Let's check our design to make sure it will work. An input value of ABCD = 0000 will select input line 0, because BCD = 000. Since A is connected to this line, its logic, in this case 0, will appear at the multiplexer output. Notice that this is the correct output according to the truth table. Likewise, an input of 0001 selects input line 1 which is connected to $\overline{A}$. Since address code A = 0 for this input value, $\overline{A}$ = 1. This produces a logic 1 at the multiplexer output. Again, this output is correct according to the original truth table on Page 3-16. Continuing in this manner, you will find that the circuit in Figure 3-12 does indeed implement the original truth table.

Our design uses the non-inverting output of the 74151. I should caution you, however, that if you use the inverted output of the 74151, the input logic to the multiplexer must be reversed to get the same result. This is also true when using a multiplexer like the 74150 which only produces an inverted output. Of course, you could always invert the respective output line to achieve the same results.

Our MSI logic circuit resulted in a two-chip design: an 8-to-1 multiplexer and an inverter for the A input variable. The same design using SSI devices would require at least four ICs. However, a multiplexer design will not always result in fewer ICs. Depending on how much the original logic equation can be simplified, a multiplexer design may or may not be more economical. For this reason, you should always compare the SSI and MSI designs before making a decision on which one to implement.

By the same token you should also examine the MSI circuit design for chip count and cost efficiency. The 74150 multiplexer design created by the obvious solution costs about twice as much as the 75151 multiplexer design just completed.

Example 3-3

Design a circuit to implement the following truth table: (Use the most economical design solution.)

A B C	y
0 0 0	0
0 0 1	1
0 1 0	1
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1

Ignoring the most significant input variable, A, there are four pairs of the variables BC as follows:

	INPUT	B C	y
Pair 0	0	0 0	0
	4	0 0	1
Pair 1	1	0 1	1
	5	0 1	0
Pair 2	2	1 0	1
	6	1 0	0
Pair 3	3	1 1	0
	7	1 1	1

Here, the value of y from the truth table has been recorded next to each pair member.

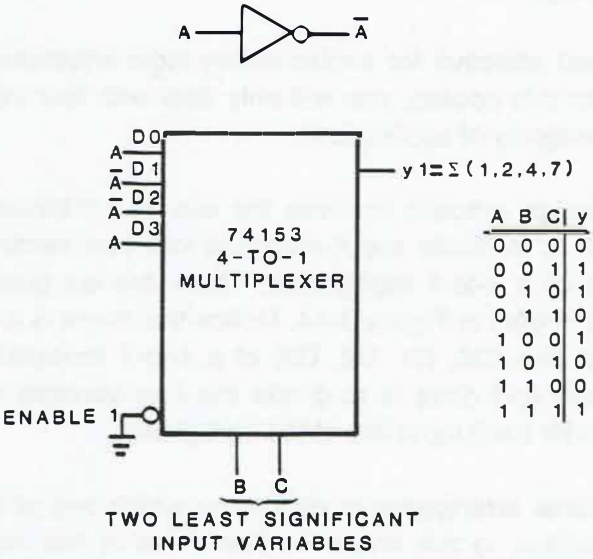
Next, assign a pair value to each pair. Recall that the pair values are assigned according to the output logic combinations for each pair as follows:

y = 00, pair value = 0
y = 01, pair value = A
y = 10, pair value = $\overline{A}$
y = 11, pair value = 1

The correct assignments for the above pairs are:

Pair 0 – A
Pair 1 – $\overline{A}$
Pair 2 – $\overline{A}$
Pair 3 – A

Since there are four pairs, a 4-to-1 multiplexer is required. So, let's use one section of a 74153. The pair values must be applied to the respective multiplexer input lines as shown in Figure 3-13. Finally, the remaining input variables, BC, are connected to the multiplexer address lines. A final check on the design will verify its correct operation.



A	B	C	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Figure 3-13

The n-1 solution for Example 3-3.

The MSI design in Figure 3-13 requires two ICs to implement: a 74153 multiplexer and a 7404 inverter. But, what about an SSI design? Take a closer look at the truth table. Do you see anything that looks familiar? I see that there is a high output each time there is an odd number of 1's on the input. This is odd parity and can be implemented directly with a single SSI device: a 7486 XOR chip. Although the multiplexer design is correct, it would be much more economical to implement the XOR design. Here is a classic example of not "seeing the forest for the trees." Did you see the XOR operation in the truth table, or did you plunge right into the forest and complete the MSI design process?

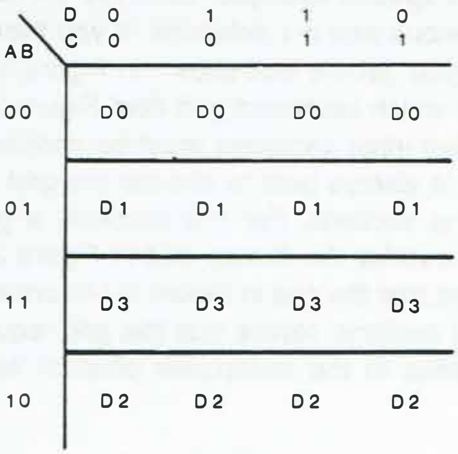
THE n-2 SOLUTION

Using the above (n-1) solution requires a 2^{n-1}-to-1 multiplexer to implement a design of n input variables. Thus, an 8-to-1 multiplexer is required to implement a design with four input variables. The size of the multiplexer can be further reduced by applying a multiplexer simplification process to the logic equation prior to implementation. This results in a design which requires a 2^{n-2}-to-1 multiplexer. As a result, with this process a four variable input can be implemented with a 4-to-1 multiplexer. The trade-off is usually a more tedious design process and additional external logic.

The n-2 solution is most effective for implementing logic equations with four or more input variables. In this course, you will only deal with four input variables, since this satisfies the majority of applications.

The n-2 multiplexer design process involves the use of a Karnaugh map. The first thing you must do is to divide the K-map grid into four sections according to the four input lines of a 4-to-1 multiplexer. There are six possible ways to slice-up the K-map, as shown in Figure 3-14. Notice that there is a 4-cell section assigned to each input line (D0, D1, D2, D3) of a 4-to-1 multiplexer. What the sectioning of the K-map grid does is to divide the four variable map into four two-variable maps, one for each input line of the multiplexer.

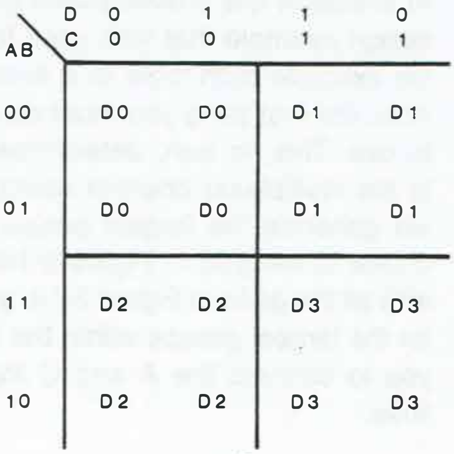
The six different sectional arrangements determine which two of the four input variables will be connected to the channel select lines of the multiplexer. For example, the grid in Figure 3-14a would require A and B to be connected to the channel select lines of a 4-to-1 multiplexer. Likewise, Figure 3-14b requires that input variables A and C must be connected to the channel select lines, and so on.



AB \ DC	0 0	1 0	1 1	0 1
00	D0	D0	D0	D0
01	D1	D1	D1	D1
11	D3	D3	D3	D3
10	D2	D2	D2	D2

ⓐ

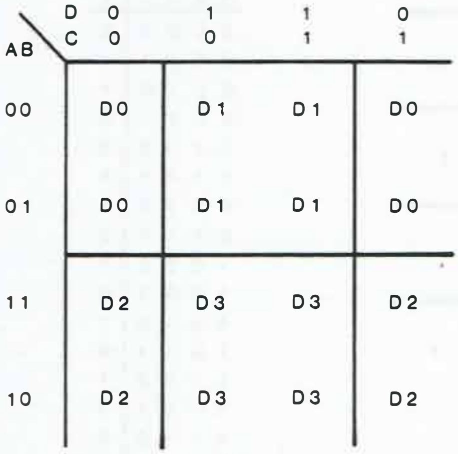
ELIMINATES A&B



AB \ DC	0 0	1 0	1 1	0 1
00	D0	D0	D1	D1
01	D0	D0	D1	D1
11	D2	D2	D3	D3
10	D2	D2	D3	D3

ⓑ

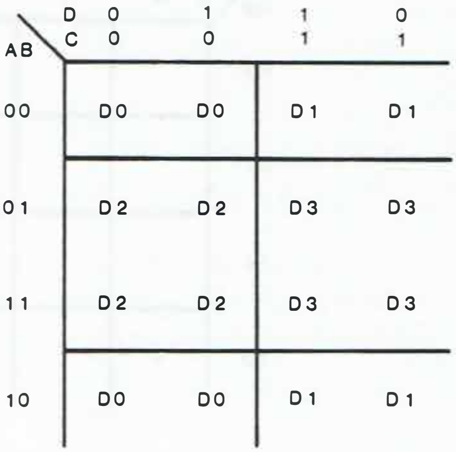
ELIMINATES A&C



AB \ DC	0 0	1 0	1 1	0 1
00	D0	D1	D1	D0
01	D0	D1	D1	D0
11	D2	D3	D3	D2
10	D2	D3	D3	D2

ⓒ

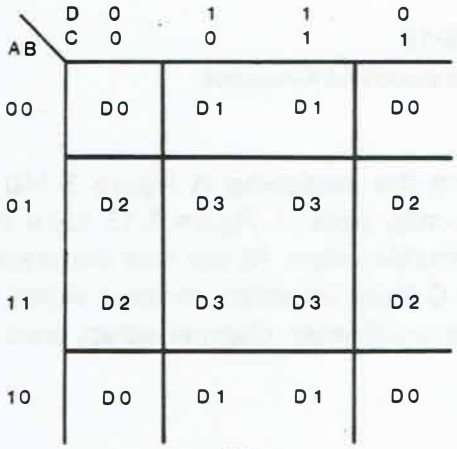
ELIMINATES A&D



AB \ DC	0 0	1 0	1 1	0 1
00	D0	D0	D1	D1
01	D2	D2	D3	D3
11	D2	D2	D3	D3
10	D0	D0	D1	D1

ⓓ

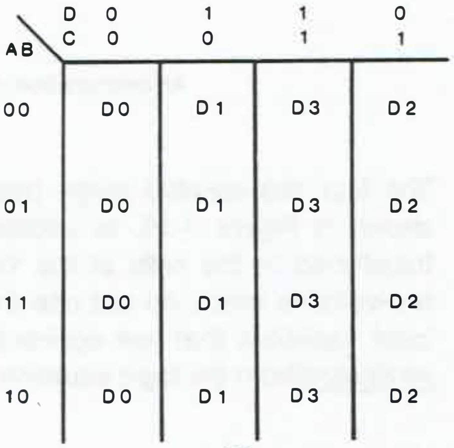
ELIMINATES B&C



AB \ DC	0 0	1 0	1 1	0 1
00	D0	D1	D1	D0
01	D2	D3	D3	D2
11	D2	D3	D3	D2
10	D0	D1	D1	D0

ⓔ

ELIMINATES B&D



AB \ DC	0 0	1 0	1 1	0 1
00	D0	D1	D3	D2
01	D0	D1	D3	D2
11	D0	D1	D3	D2
10	D0	D1	D3	D2

ⓕ

ELIMINATES C&D

Figure 3-14

There are six possible ways to section a 4-variable K-map grid when using the n-2 multiplexer design solution.

At this point it is probably best to look at a specific example. Let's use the same design example that was used for the obvious and n-1 solutions. If you transfer the example truth table to a K-map grid, you get the plot shown in Figure 3-15. Now, the first thing you must do is decide which sectioned grid from Figure 3-14 to use. This, in turn, determines which two input variables must be connected to the multiplexer channel select lines. It is always best to choose the grid that will generate the largest groups within the sections. For this problem, a good choice is the grid in Figure 3-14b. If you overlay the K-map plot in Figure 3-15 with all the grids in Figure 3-14, you will find that the one in Figure 3-14b provides for the largest groups within the individual sections. Notice that this grid requires you to connect the A and C input variables to the multiplexer channel select lines.

	D	0	1	1	0
AB	C	0	0	1	1
00			1		1
01			1		1
11		1	1		
10		1			1

A	B	C	D	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Figure 3-15

An example truth table and its associated K-map plot.

The four two-variable maps obtained from the sectioning in Figure 3-14b are shown in Figure 3-16. In addition, the K-map plots in Figure 3-15 have been transferred to the cells of the four two-variable maps. Notice that the resulting two-variable maps do not use the A and C input variables. In other words, the input variables that are connected to the multiplexer channel select lines are eliminated from the logic equation.

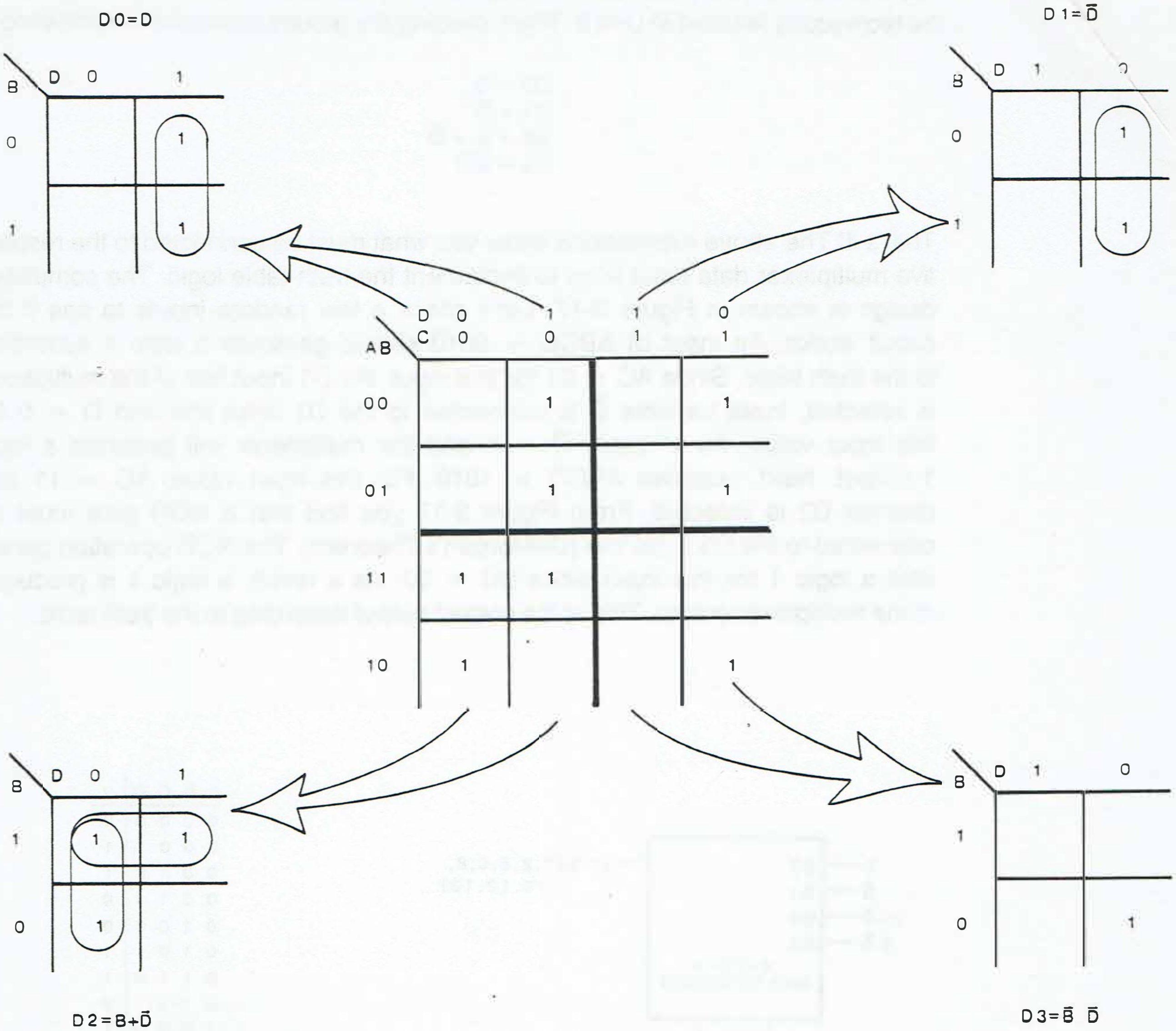


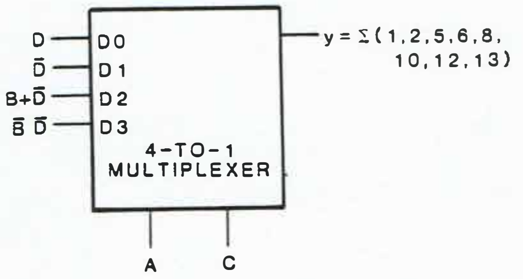
Figure 3-16

The four 2-variable maps are broken out from the 4-variable map.

The next thing you must do is group the two-variable maps as shown, using the techniques learned in Unit 2. Then, reading the groups produces the following:

$$\begin{aligned} D0 &= D \\ D1 &= \overline{D} \\ D2 &= B + \overline{D} \\ D3 &= \overline{B}\,\overline{D} \end{aligned}$$

That's it! The above expressions show you what must be connected to the respective multiplexer data input lines to implement the truth table logic. The completed design is shown in Figure 3-17. Let's check a few random inputs to see if the circuit works. An input of ABCD = 0010 should generate a logic 1 according to the truth table. Since AC = 01 for this input, the D1 input line of the multiplexer is selected. Input variable $\overline{D}$ is connected to the D1 input line and D = 0 for this input value. As a result, $\overline{D}$ = 1 and the multiplexer will generate a logic 1 output. Next, suppose ABCD = 1010. For this input value, AC = 11 and channel D3 is selected. From Figure 3-17 you find that a NOR gate must be connected to the D3 input line (DeMorgan's Theorem). The NOR operation generates a logic 1 for this input since BD = 00. As a result, a logic 1 is produced at the multiplexer output. This is the correct output according to the truth table.


D
$\overline{D}$
$B+\overline{D}$
$\overline{B}\,\overline{D}$
D0
D1
D2
D3
4-TO-1
MULTIPLEXER
$y = \Sigma(1,2,5,6,8,10,12,13)$
A
C

A	B	C	D	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Figure 3-17

Implementation of a 4-variable truth table using the n-2 solution and a 4-to-1 multiplexer.

The design in Figure 3-17 could be implemented with two ICs: a 74153 multiplexer and a 7402 quad NOR chip as shown in Figure 3-18. Two of the NOR gates in the 7402 are used as inverters. Let's compare this design to the n-1 design (Figure 3-12). Recall that the n-1 design also required two chips: a 74151 multiplexer and an inverter which could be implemented using a 7402 NOR chip. So, the major difference in the two designs is between the 74151 and 74153 multiplexers. Actually, the two designs cost about the same to implement since a 74151 and a 75153 sell for about the same price. However, with the n-2 design, there is an extra 4-to-1 multiplexer available in the 74153 to implement another design. This may or may not be an advantage, depending on the application.

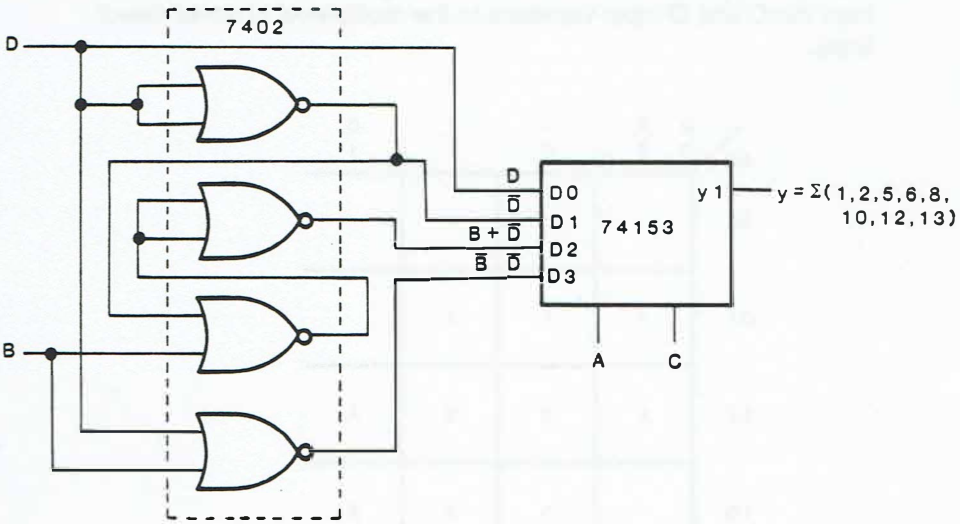


Figure 3-18

The design in Figure 3-17 is implemented with two chips: a 74153 4-to-1 multiplexer and a 7402 quad NOR.

Example 3-4

Design a multiplexer circuit to implement the logic required to illuminate the "e" segment of a common anode 7-segment LED.

This is the problem that you solved with an SSI design through a series of examples in Unit 2. Let's see if you can obtain a simpler MSI design using multiplexers.

The K-map obtained in Unit 2 is shown again in Figure 3-19. Comparing this map to the sectional grids in Figure 3-14, you find that the grid in Figure 3-14f results in the best grouping within the individual sections. The use of this grid requires you to connect the C and D input variables to the multiplexer channel select lines.

AB \ DC	00	10	11	01
00		1	1	
01	1	1	1	
11	X	X	X	X
10		1	X	X

Figure 3-19
The K-map for the "e" segment problem.

The two-variable sections of Figure 3-14f are shown in Figure 3-20 with the corresponding plots transferred from the "e" segment K-map. Grouping the plots and reading the groups you get:

D0 = B
D1 = 1
D2 = 0
D3 = 1

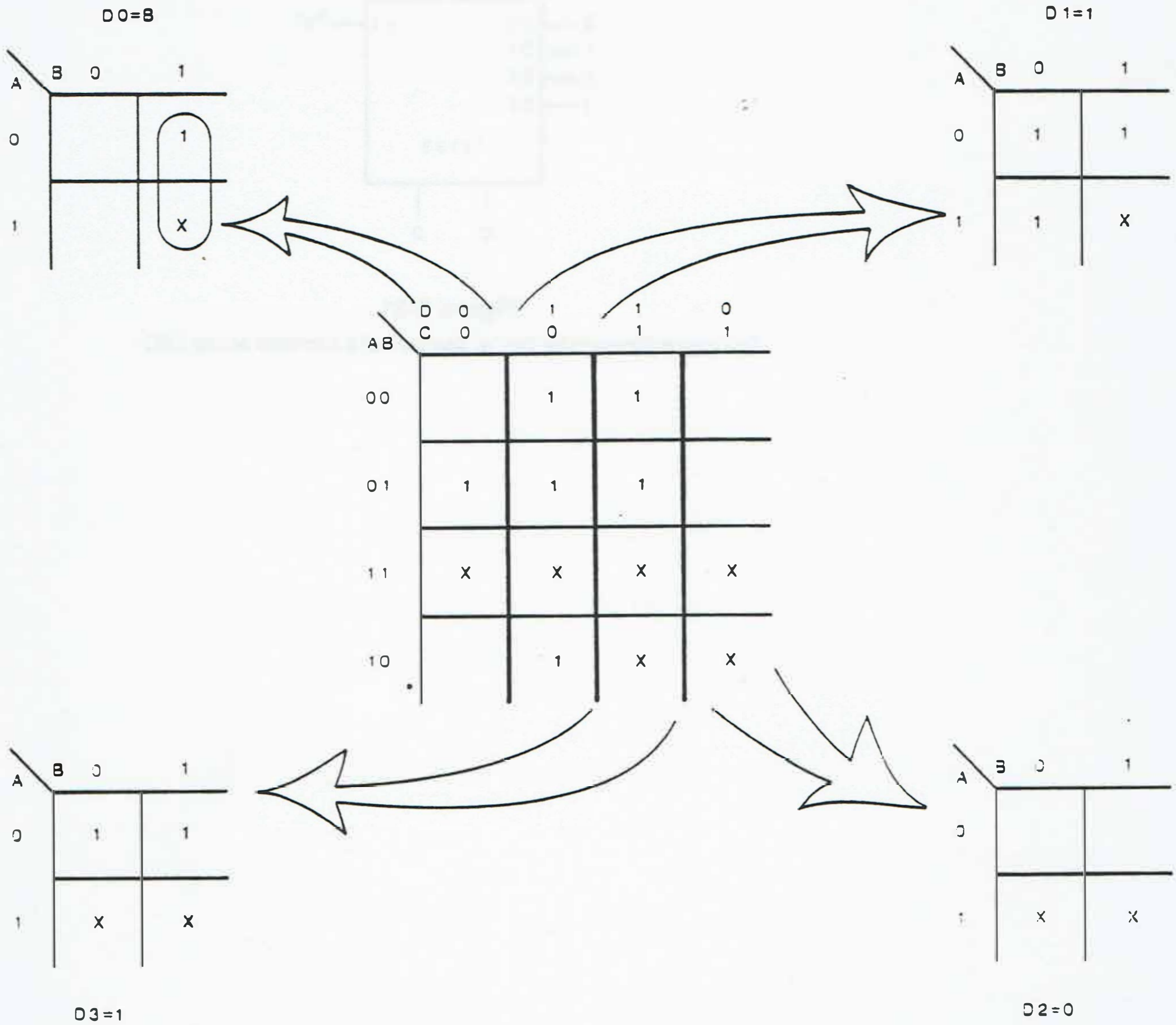


Figure 3-20

The four 2-variable maps generated by the column grid in Figure 3-14f.

The resulting design is shown in Figure 3-21. This design requires a single 74153 multiplexer. The SSI design you obtained in Unit 2 required a single 7400 NAND chip. The 74153 is about twice the cost of a 7400. However, the remaining 4-to-1 multiplexer in the 74153 could be used to implement the logic for another one of the LED segments.

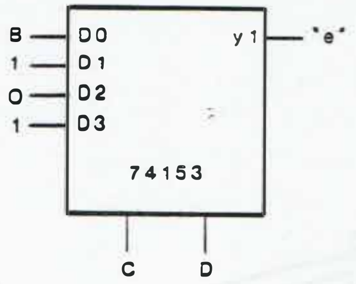


Figure 3-21

The final n-2 design for the "e" segment of a common anode LED.

Self-Test Review

1. Another name for a multiplexer is a ________ ____________.

2. How many channel select lines are required for a 32-to-1 multiplexer? ________________.

3. Design a 16-to-1 multiplexer using 4-to-1 multiplexers.

4. How many ICs are required to implement your design in problem 3? ____

5. Can the 16-to-1 multiplexer in problem 3 be implemented with fewer ICs? If so, how? ______________________________

6. Design a multiplexer circuit to detect when any two of four inputs are high. Use the obvious solution design procedure with a multilexer that generates an inverted output.

7. How many of what type ICs are required to implement the design in problem 6? ____________________

8. Redesign the circuit in problem 6 using the n – 1 solution with a multiplexer that generates a non-inverting output.

9. How many of what type ICs are required to implement the design in problem 8? ____________________

10. Redesign the circuit in problem 6 using the n – 2 solution with a multiplexer that generates a non-inverting output.

11. How many of what type ICs are required to implement the design in problem 10? ________________

12. Using a recent component catalog, look up the cost of implementing the above three designs. Which design is the most cost effective?

____

Answers

1. data selector

2. Five channel select lines since $2^5 = 32$.

3. See Figure 3-22 on Page 3-38.

4. Three 74153 dual 4-to-1 multiplexer ICs.

5. Yes, a single 74150.

6. See Figure 3-23 on Page 3-39.

7. A single 74150 16-to-1multiplexer IC. However, the input logic must be inverted, since the 74150 has an inverted output.

8. See Figure 3-24 on Page 3-39.

9. A single 74151 8-to-1 multiplexer IC and a 7404 inverter IC.

10. See Figure 3-25 on Page 3-40. Here, the grid in Figure 3-14a was used to eliminate A and B. Consequently, A and B must be connected to the address inputs of the multiplexer. Because of the nature of the groupings, any of the six possible K-maps would have produced an acceptable solution. Ideally, you would select the K-map that produced the largest groupings.

11. A single 74153 4-to-1 multiplexer IC, a 7486 quad 2-bit XOR IC, and a 7402 quad 2-bit NOR IC.

12. As of this writing, the n-1 design produces the most cost effective circuit. The n-2 design could be more cost effective if the unused 4-to-1 multiplexer in the 74153 were used to implement another logic function.

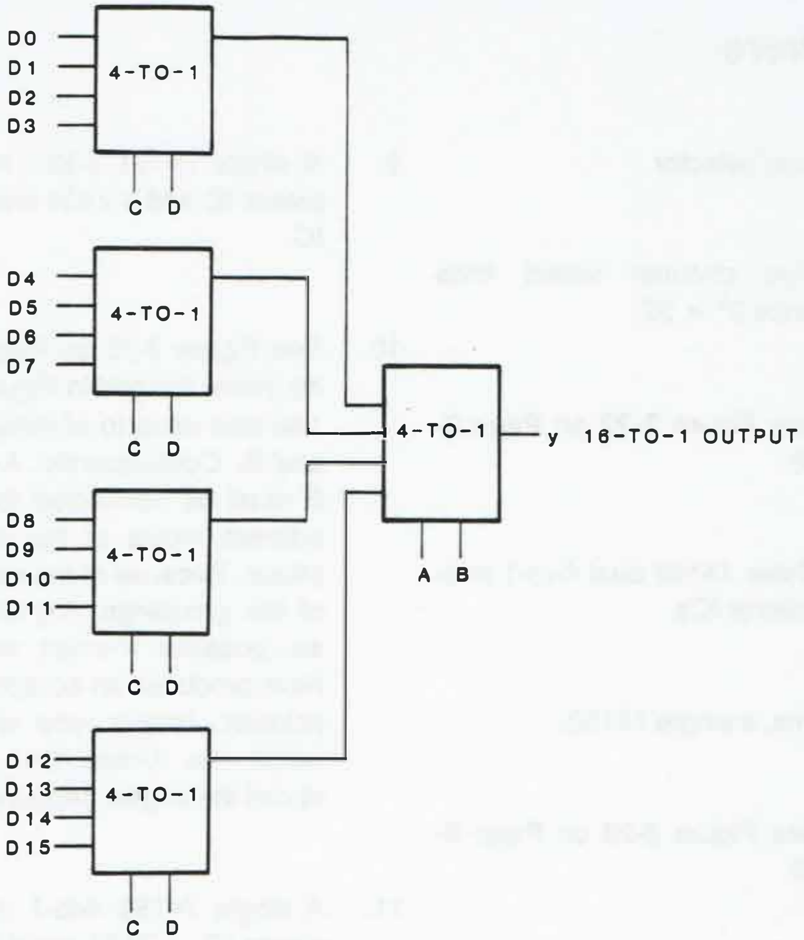
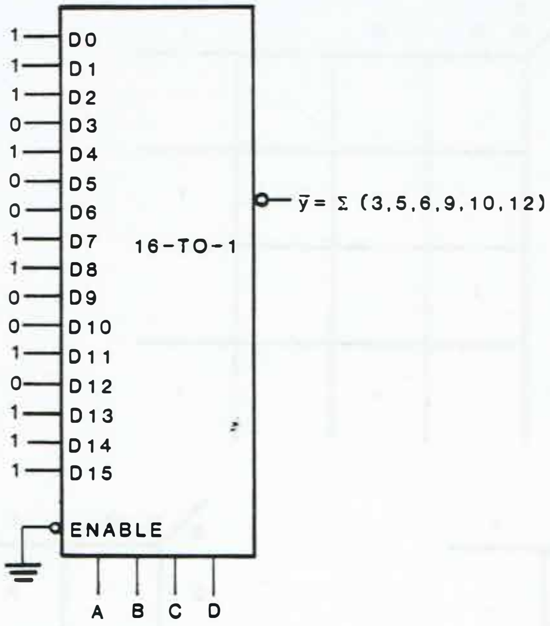


Figure 3-22

A 16-to-1 multiplexer for problem 3.



A	B	C	D	$\bar{y}$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Figure 3-23

The obvious solution for problem 6.

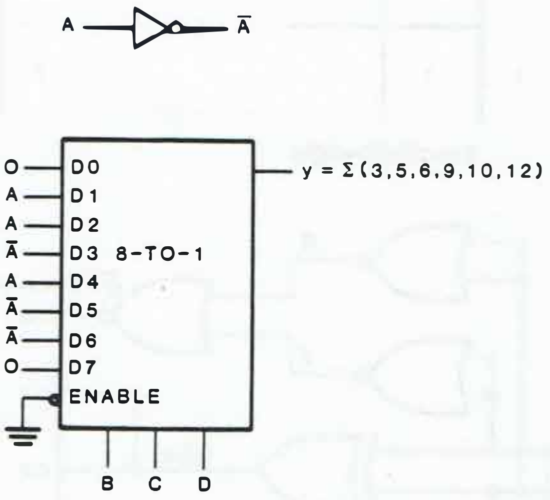


Figure 3-24

The n – 1 solution for problem 8.

AB \ DC	00	10	11	01
00			1	
01		1		1
11	1			
10		1		1

C \ D	0	1
0		
1		1

D0=CD=$\overline{\bar{C}+\bar{D}}$

C \ D	0	1
0		1
1	1	

D1=C$\bar{D}$+$\bar{C}$D=C⊕D

C \ D	0	1
0		1
1	1	

D2=C$\bar{D}$+$\bar{C}$D=C⊕D

C \ D	0	1
0	1	
1		

D3=$\bar{C}\bar{D}$=$\overline{C+D}$

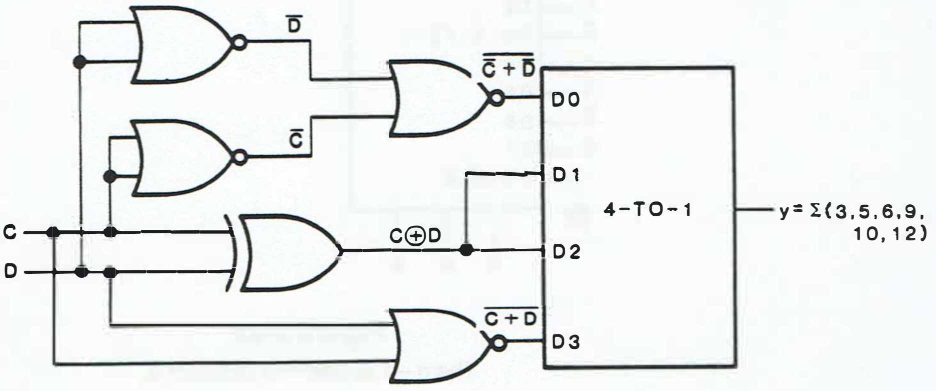


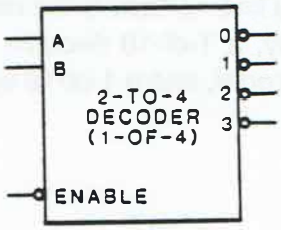
Figure 3-25
The n-2 solution for problem 10.

DECODERS

The second MSI device that you must learn about is the decoder. Decoders can also be used to implement combinational logic. They are extremely economical to use when more than one logic equation must be implemented. Recall from our discussion of multiplexers that a single multiplexer could only be used to implement a single logic equation. Additional logic equations require additional multiplexers for their implementation. On the other hand, a single decoder can be used to implement several logic equations. As always there is a price to pay. You will soon discover that the trade-off in using a decoder to implement multiple logic equations is extra external logic.

Decoder Types

A decoder is simply a logic circuit that is designed to **detect** the presence of a specific binary input. The black box decoder in Figure 3-26 illustrates this idea. This decoder has two input lines, A and B, and four output lines: 0, 1, 2, and 3. For this reason it is called a **2-line-to-4-line** decoder.



A	B	0	1	2	3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Figure 3-26

A decoder is a logic detection circuit.

The decoder in Figure 3-26 has four possible values for the two input lines: 00, 01, 10, and 11. It detects when any one of these input values is present by generating a logic 0 on the corresponding output line. For example, when an input value of 00 is present, output line 0 goes low while the other output lines remain high. Likewise, when a value of 01 is present at the input, a logic 0 is generated on line 1 while lines 0, 2, and 3 remain high. In the same way, output line 2 goes low for an input value of 10, and line 3 goes low for an input value of 11.

The associated truth table in Figure 3-26 summarizes the operation of this decoder. First, notice that a logic 0 is the active output state. This is typical of most decoders since, recall from Unit 1 that it is easier to sink current than it is to source it within an IC. Second, by definition, only one of the four output lines is active for any given input. For this reason, another name for this device is a **1-of-4** decoder.

Other popular decoders are 1-of-8, 1-of-10, and 1-of-16. Each of these decoders are shown in Figure 3-27 along with their respective truth tables. In each case, the idea is the same: a given input is detected by the decoder generating a logic 0 on the corresponding output line.

The 1-of-8 decoder in Figure 3-27a can be thought of as a **binary-to-octal** decoder, since the active output line number is the octal equivalent of a given binary input value. In the same way, a 1-of-10 decoder in Figure 3-27b is sometimes called a **BCD-to-decimal** decoder, and a 1-of-16 decoder in Figure 3-27c is called a **binary-to-hex** decoder.

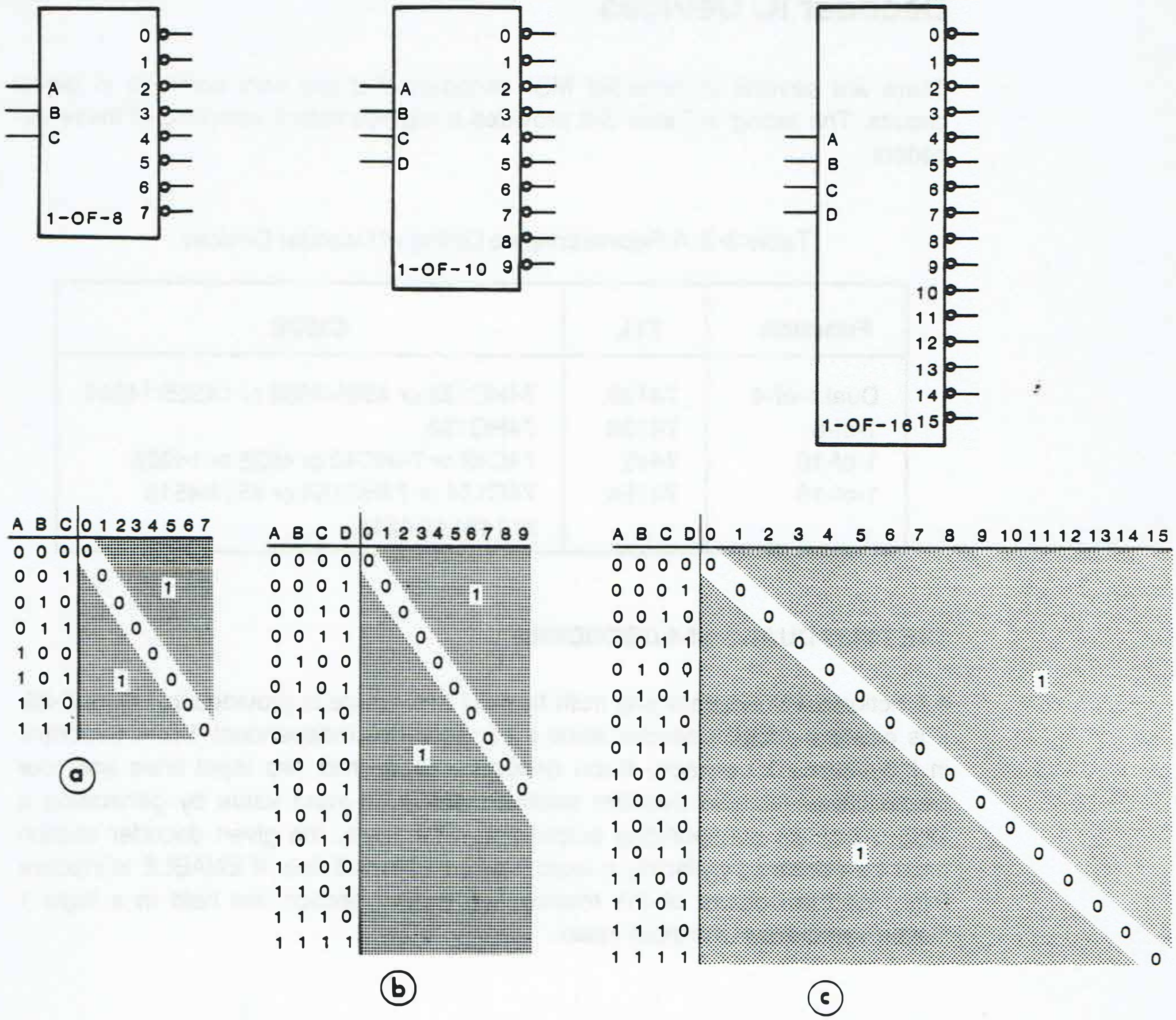


Figure 3-27

Three decoder sizes: 1-of-8 (a), 1-of-10 (b), and 1-of-16 (c).

Decoder IC Devices

There are several commercial MSI decoders that are very common in digital circuits. The listing in Table 3-2 provides a representative sampling of these decoders.

Table 3-2: A Representative Listing of Decoder Devices

Function	TTL	CMOS
Dual 1-of-4	74139	74HC139 or 4555/4556 or 14555/14556
1-of-8	74138	74HC138
1-of-10	7442	74C42 or 74HC42 or 4028 or 14028
1-of-16	74154	74C154 or 74HC154 or 4514/4515 or 14514/14515

THE 74139 DUAL 1-of-4 DECODER

A functional I/O diagram and truth table of this device is provided in Figure 3-28. It is called a "dual" decoder since it contains two independent 1-of-4 decoders in the same IC package. Each decoder section has two input lines and four output lines. A given decoder section detects an input value by generating a logic 0 on its corresponding output line. Of course, the given decoder section must be enabled by placing a logic 0 on its ENABLE line. If ENABLE is inactive (high) all the outputs of the respective decoder section are held in a logic 1 state, regardless of the input value.

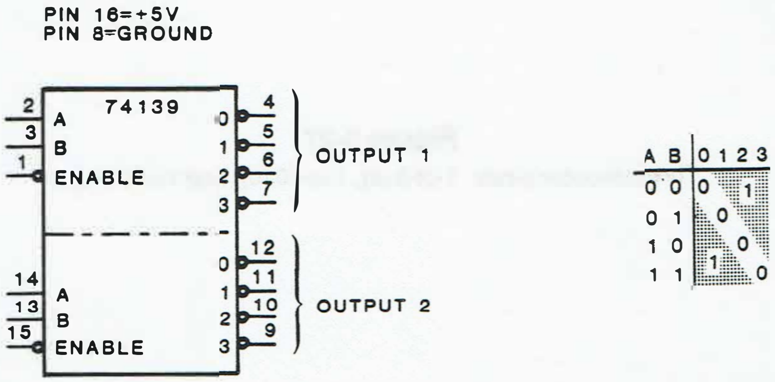


Figure 3-28

The 74139 dual 1-of-4 decoder.

THE 74138 1-of-8 DECODER

The 74138 shown in Figure 3-29 is straight forward. The decoder detects one of eight input values and generates a logic 0 on the corresponding output line. The only feature unique to the 74138 is its three enable lines: $\overline{E1}$, $\overline{E2}$, and E3. All of these lines must be active for the decoder to operate. Notice that $\overline{E1}$ and $\overline{E2}$ are active low, while E3 is active high. For continuous operation, you can hardwire $\overline{E1}$ and $\overline{E2}$ to ground, and E3 to the supply voltage. If any one of the enable lines are inactive, all the decoder output lines are held in a logic 1 state.

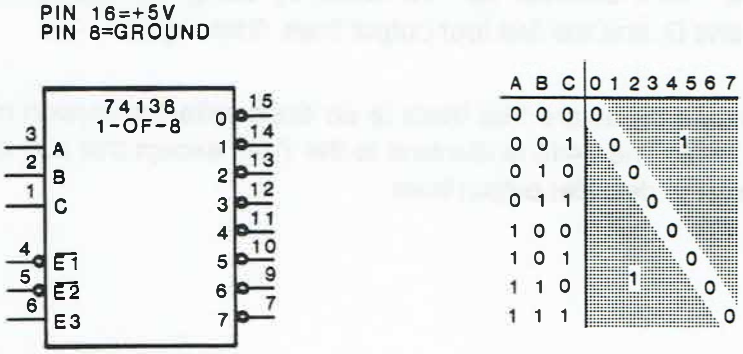


Figure 3-29

The 74138 1-of-8 decoder.

Example 3-5

Construct a 1-of-4 decoder using a 74138 1-of-8 decoder.

The 74138 1-of-8 decoder in Figure 3-29 will operate as a 1-of-4 decoder when you use only the two least significant input lines, B and C, and the first four output lines, 0 through 3. Of course, the enable lines must be active for any operation.

THE 7442 1-of-10 DECODER

The 7442 shown in Figure 3-30 has four input lines, A, B, C, and D, and ten output lines, 0 through 9. As the input values range from 0000 to 1001, the decoder generates a logic 0 on the corresponding output line. Notice from the truth table that any value above 1001 is not detected. For this reason, the 7442 is sometimes called a BCD-to-decimal decoder.

You can make a 1-of-8 decoder out of the 7442 by using only the three least significant input lines, B, C, and D, and the first eight output lines, 0 through 7. Likewise, a 1-of-4 decoder can be made by using the two least significant input lines, C and D, and the first four output lines, 0 through 3.

Finally, you should be aware that there is an open collector version of the 7442 available, the 7445. The 7445 is identical to the 7442 except that pull-up resistors must be used on the decoder output lines.

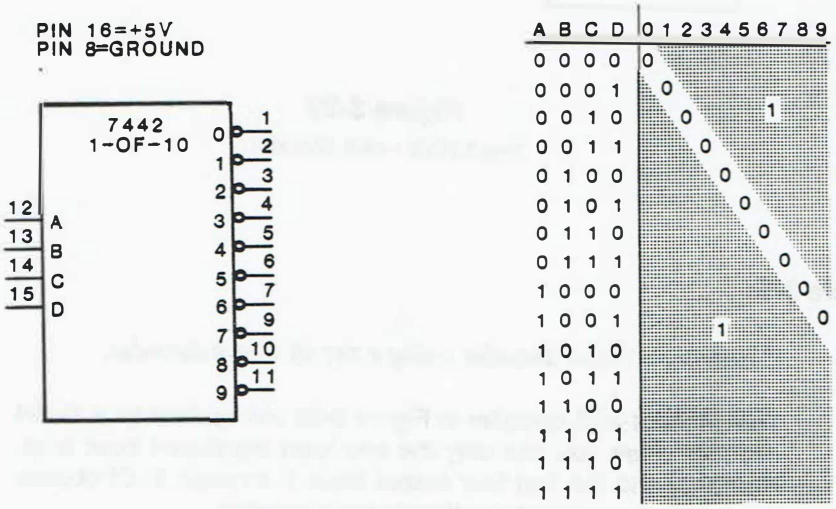


Figure 3-30

The 7442 1-of-10 decoder.

THE 74154 1-of-16 DECODER

This last decoder is the largest that is commercially available. As you can see from Figure 3-31, the 74154 generates a logic zero on one of its sixteen output lines when it detects any 4-bit input value from 0000 through 1111. By using fewer input and output lines, you can operate the 74154 as a 1-of-2, 1-of-4, 1-of-8, or 1-of-10 decoder.

The 74154 has two enable lines, $\overline{E1}$ and $\overline{E2}$, both of which are active low. A high level on either of these lines causes all output lines to go high, regardless of the input logic.

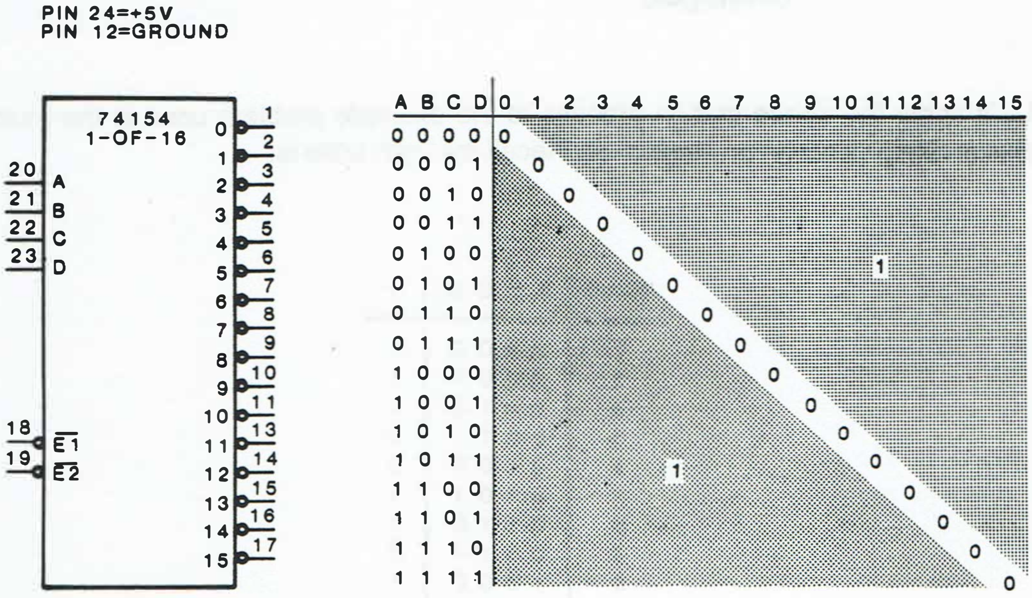


Figure 3-31
The 74154 1-of-16 decoder.

Implementing Logic Equations with Decoders

Like multiplexers, decoders can be used to implement logic equations. The advantage of using a decoder over a multiplexer is that a single decoder can implement several equations. The disadvantage is that decoders always require the use of external SSI logic gates.

The design process is relatively easy and straight forward. Just follow the steps below and you can't go wrong.

Step 1: From a truth table or existing logic equation, write an SOP expression in **numerical form**.

Step 2: Pick a decoder of sufficient size to implement the logic equation.

Step 3: Connect the circuit input variables to the respective input lines of the decoder.

Step 4: For each numerical value in the SOP expression, connect the decoder output line with the same number to the input of a NAND gate.

Let's apply the above design process to the example problem used in the multiplexer design section on Page 3-16. Recall the truth table is:

INPUT	A B C D	y
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	1
3	0 0 1 1	0
4	0 1 0 0	0
5	0 1 0 1	1
6	0 1 1 0	1
7	0 1 1 1	0
8	1 0 0 0	1
9	1 0 0 1	0
10	1 0 1 0	1
11	1 0 1 1	0
12	1 1 0 0	1
13	1 1 0 1	1
14	1 1 1 0	0
15	1 1 1 1	0

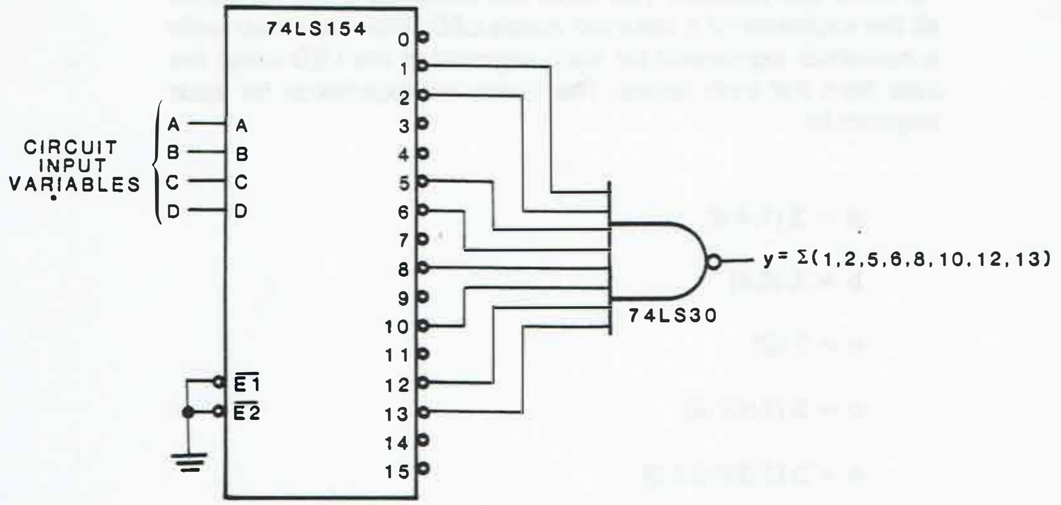
The first step of the process requires you to write an SOP expression in numerical form. The expression obtained for this truth table is $y = \Sigma\,(1,2,5,6,8,10,12,13)$.

Next, you must pick a decoder of sufficient size to implement the above equation. The listing in Table 3-3 shows you what size decoder is required to implement a logic equation containing two, three, or four input variables. From here, you find that a 1-of-16 decoder is required. Let's use a 74LS154.

Table 3-3: Decoder Sizes to Implement 2, 3, or 4 Input Variables

Number of Input Variables	**Decoder Required**
Two	1-of-4
Three	1-of-8
Four (Straight Binary)	1-of-16
Four (Binary Coded Decimal)	1-of-10

In the third step, you must connect the circuit input variables to the decoder input lines as shown in Figure 3-32.



A	B	C	D	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Figure 3-32

Implementing a logic function with a decoder and a NAND gate.

Finally, you must connect each decoder output line in the equation to the input of a NAND gate. Here, decoder output lines 1, 2, 5, 6, 8, 10, 12, and 13 are connected to the input of a 74LS30 8-bit NAND gate. That's all there is to it!

Now, let's check our design. For an input value of 0000 the circuit should generate a logic 0 per the truth table. An input value of 0000 to the decoder is detected by output line 0 of the decoder. Thus, output line 0 goes low and all the remaining output lines are high. Since line 0 does not drive the NAND gate, all the input lines to the gate are high. Consequently, the gate output is low.

Next, suppose a value of 0001 is applied to the circuit. According to the truth table, the circuit should produce a logic 1. An input value of 0001 causes output line 1 of the decoder to go low. This, in turn, causes the NAND gate output to go high regardless of the other logic, since any logic 0 input to a NAND gate generates a logic 1 output from the gate. Should you check the remaining input values, you will find that our decoder design does indeed implement the truth table.

Example 3-6

Design a decoder circuit using a single decoder to implement all the segment equations for a common anode 7-segment LED.

To solve this problem, you must first construct a truth table for all the segments of a common anode LED. Then you must write a numerical expression for each segment of the LED using the data from the truth tables. The numerical expression for each segment is:

$$a = \Sigma\,(1,4,6)$$

$$b = \Sigma\,(5,6)$$

$$c = \Sigma\,(2)$$

$$d = \Sigma\,(1,4,7,9)$$

$$e = \Sigma\,(1,3,4,5,7,9)$$

$$f = \Sigma\,(1,2,3,7)$$

$$g = \Sigma\,(0,1,7)$$

The above expressions were obtained by first constructing a truth table for all of the segments of a common anode LED. The logic equation for each segment was then written in numerical form.

Next, a 1-of-10 decoder must be used since we are dealing with binary coded decimal inputs (see Table 3- 3).

The four circuit input variables A, B, C, and D are then connected directly to the A, B, C, and D inputs of the 74LS42 1-of-10 decoder.

Finally, the decoder outputs are used to drive NAND gates as shown in Figure 3-33. The entire circuit can be implemented with four chips: a 74LS42 decoder, 74LS20 dual 4-bit NAND, 74LS10 triple 3-bit NAND, and a 74LS00 quad 2-bit NAND. The inverters have been implemented using two extra NAND gates in the 74LS00 chip. Try that with an SSI design!

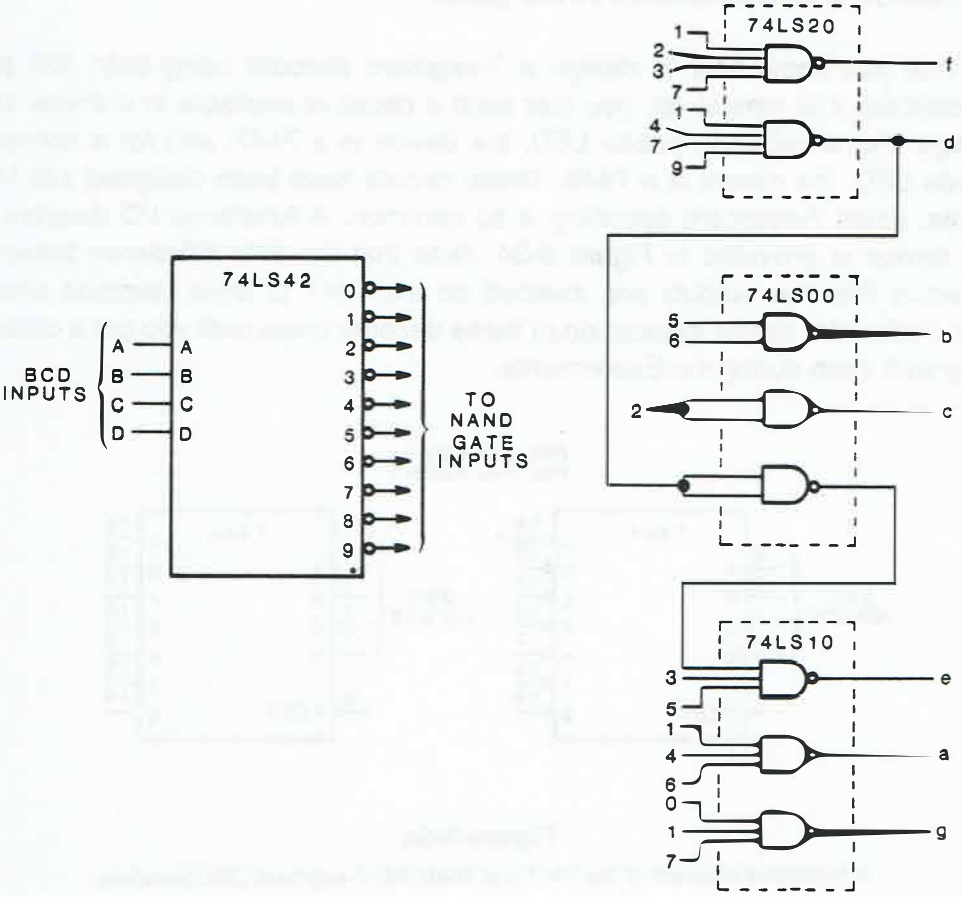


Figure 3-33

A decoder implementation of all segments for a common anode 7-segment LED. (Interconnecting lines left out for clarity.)

> Notice that I included one little trick to save an IC. Since four of the terms of the "d" segment are also used for the "e" segment, I have connected the "d" segment output to one of the input lines of the "e" segment NAND via an inverter. The inverter is required because the "d" segment NAND gate inverts the truth table logic. You might want to check the circuit against a common anode LED truth table to assure that it will work.

The important lesson to learn from this example, is that a single decoder can be used to implement several logic equations. However, as you can see, several external NAND gates are required. Coincidentally, four 74153 multiplexers can be used to implement the same design using the n-2 solution. But, the four 74153 multiplexers plus the associated inverters would be more costly than the single 7442 decoder and its associated NAND gates.

Now that you know how to design a 7-segment decoder using both SSI and MSI devices, it is time to tell you that such a circuit is available in a single MSI package. For a common anode LED, the device is a 7447; and for a common cathode LED, the device is a 7448. These circuits have been designed into MSI devices, since 7-segment decoding is so common. A functional I/O diagram of each device is provided in Figure 3-34. Note that the only difference between the two is that the outputs are inverted on the 7447 to drive common anode LEDs. I will defer further discussion of these decoder chips until you get a chance to play with them during the Experiments.

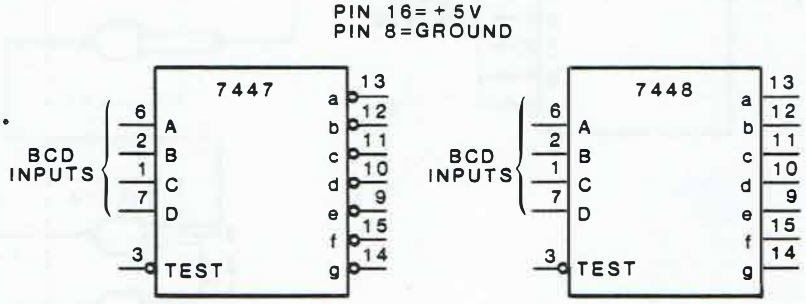


Figure 3-34

A functional diagram of the 7447 and 7448 MSI 7-segment LED decoders.

Address Decoding

One of the most common applications for MSI decoders is in computer circuits for address decoding. All computers use a **bus structure** for transferring binary information within the system. As shown in Figure 3-35a, a **bus** is nothing more than a set of parallel signal lines (conductors) used to transmit and receive binary information. Each signal line within the bus is assigned to one bit within a multi-bit word. Thus, an 8-bit bus is used to transfer 8-bit, or **byte** size, words. A 16-bit bus is used to transfer 16-bit words, and so forth.

For diagramming clarity, all the individual lines of a bus are not normally shown. If they were, a computer schematic would look like a "rat's nest." Rather, broad arrows are used to represent the bus as shown in Figures 3-35b and 3-35c. The number within the arrow represents the size of the bus. In addition, a given bus can be unidirectional or bidirectional, as indicated by the arrow directions. Unidirectional buses are obviously one-way signal paths, while bidirectional buses are two-way.

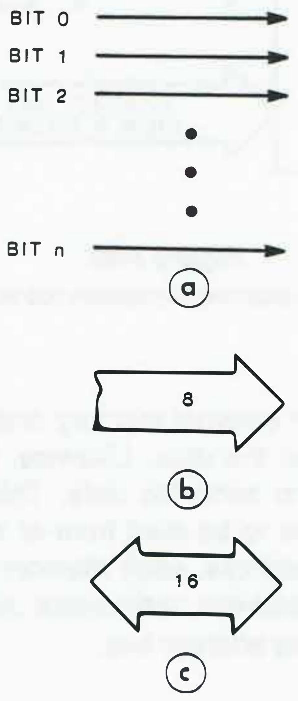


Figure 3-35

Buses are a set of parallel conductors used to transfer binary information (a). Broad arrows are used to represent unidirectional (b) and bidirectional (c) buses.

Buses within a digital system are divided according to their function within the system. In general, any digital computer system contains three functional buses: an **address bus**, a **data bus**, and a **control bus**. These bus functions are illustrated by Figure 3-36. This figure shows the fundamental bus structure of a typical microprocessor system. The idea is to transfer information between the microprocessor unit, or MPU, and external memory and peripheral I/O units using the data bus. The information to be transferred is the **data** used or generated by the MPU during the execution of a program. Therefore, the binary signals on the data bus tell you **what** information is being transferred. Data is received by the MPU during a **read** operation, and transmitted from the MPU during a **write** operation. Thus, to perform read/write operations the data bus must be a bidirectional bus as shown.

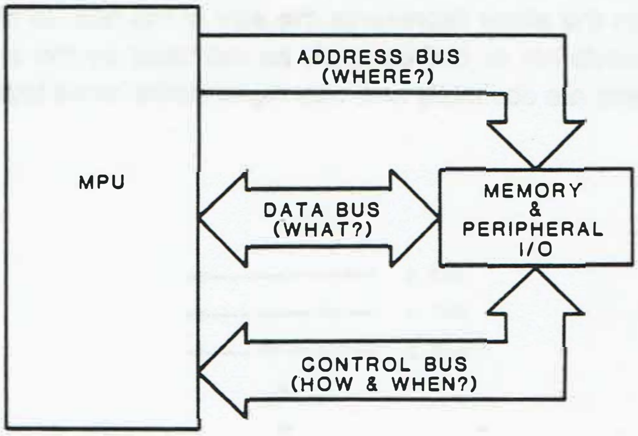


Figure 3-36
A typical microprocessor system bus structure.

When data is to be read from external memory and peripheral I/O devices, the MPU must know where to get the data. Likewise, when data is to be written, the MPU must know where to send the data. This is the job of the address bus. It indicates **where** data is to be read from or written to within the system. Just as your house has an address, each memory location and peripheral I/O device in the system has an address assignment. Addresses are always generated by the MPU on the one-way address bus.

Finally, the control bus provides information to and from the MPU about **how** and **when** to transfer data. Most of this control information is associated with the timing of operations within the system.

So, you're probably wondering **where** decoders fit into all of this? (Hint: think about the boldface word in the above sentence.) The major application of decoders in any computer system is for decoding addresses on the address bus. A typical microprocessor can generate 65,536, or 64K, addresses. Now, let's fantasize a "bit". Suppose that **you** are a read/write memory location and one of these addresses has been assigned to you. The MPU performs a data transfer operation and places an address on the address bus to indicate what memory location the data is to be read from or written to. This creates a major dilemma. You must know if the address on the address bus is yours so that you can send or receive the data via the data bus. This is the job of an **address decoder**: to signal you when your address appears on the bus.

Granted, the above analogy is an over simplification, but it clearly illustrates the function of an address decoder in a computer system. The decoder must detect a given address or set of addresses on the address bus, and signal the proper memory and/or peripheral I/O devices for the data transfer operation.

As stated earlier, most microprocessors are capable of generating 64K addresses. They do this using a 16-bit address bus ($2^{16} = 64K$). As a result, to detect a given address, a decoder must be designed to recognize one of 65,536 possible addresses. In other words, the design task requires a 1-of-64K decoder. Since such a device is not commercially available, you must combine smaller MSI decoders to accomplish the task.

A 1-of-64K decoder can be designed using four 1-of-16 decoders as shown in Figure 3-37. (Note: 64/16 = 4) Here, observe that the 16-bit address bus lines are labeled A0 through A15 and have been divided-up into four groups of four bits each. Each group of four bits is assigned to the input of a single decoder as shown.

To decode a given address you first need to convert the binary address to hexadecimal. This becomes apparent if you recall that a 1-of-16 decoder is a binary-to-hex decoder. Once you have the hexadecimal equivalent of the address, simply connect the corresponding decoder output lines to the input of a 4-bit NOR gate.

For example, suppose you wish to decode **decimal** address 4660. The MPU generates the binary equivalent of this, or 0001 0010 0011 0100, on its sixteen address bus lines A0 through A15. The hexadecimal equivalent of this value is 1234. Consequently, when this address is present, the most significant decoder generates a logic 0 on output line 1, the next decoder generates a logic 0 on output line 2, the third decoder activates line 3, and the least significant decoder activates line 4. If these decoder output lines are used to drive a 4-bit NOR gate, the gate output will only be high when hexadecimal address 1234 is present on the address bus. For all other addresses, the NOR gate output will be low. Of course, the output logic could be reversed by adding an inverter to the NOR gate output as shown by the dotted lines in the figure.

It is important to note that the four decoders in Figure 3-37 can be used to decode all of the 64K addresses by simply NORing different decoder outputs. This is called **complete decoding**, since all of the address bus lines are being decoded. The problem with complete decoding is that a 4- bit NOR gate is required for each address to be decoded. For this reason it becomes impractical to decode too many addresses. To save logic, a technique called **partial decoding** is employed when several hundred or thousand addresses must be decoded. Partial decoding uses the same idea as complete decoding, but not all of the address bus lines are decoded. A more detailed discussion of decoding must be left to a course in Microprocessor Design theory. For now, it is only important that you get a "feel" for how decoders are used in computer bus systems.

Using Decoders as Demultiplexers

An MSI decoder is often referred to as a **demultiplexer**, since it can be made to operate as a **data distributor**. A demultiplexer, or data distributor, is simply the opposite of a multiplexer, or data selector. Recall that a multiplexer is simply an electronic switch that selects one of several binary inputs and routes it to a single output. Conversely, a demultiplexer is an electronic switch that takes a single binary input and distributes it to one of many outputs. This idea is illustrated by the 1- of-4 demultiplexer in Figure 3-38. The associated truth table shows how an address code is used to distribute the input data, D_{in}, to one of four outputs: 0, 1, 2, or 3. Notice that the data input logic will appear on the selected output line.

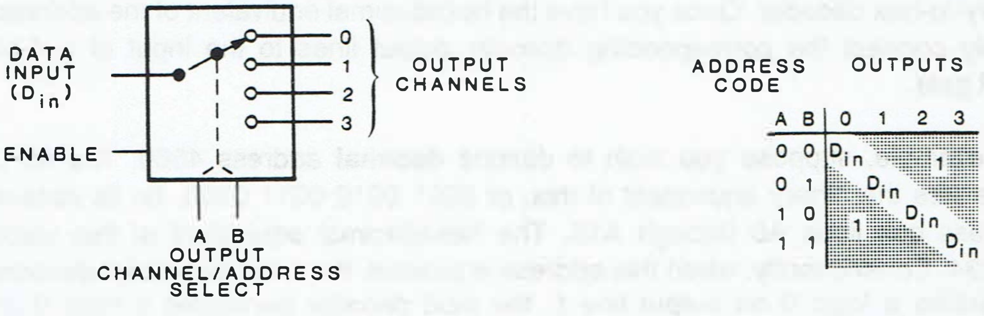


Figure 3-38

A 1-of-4 demultiplexer.

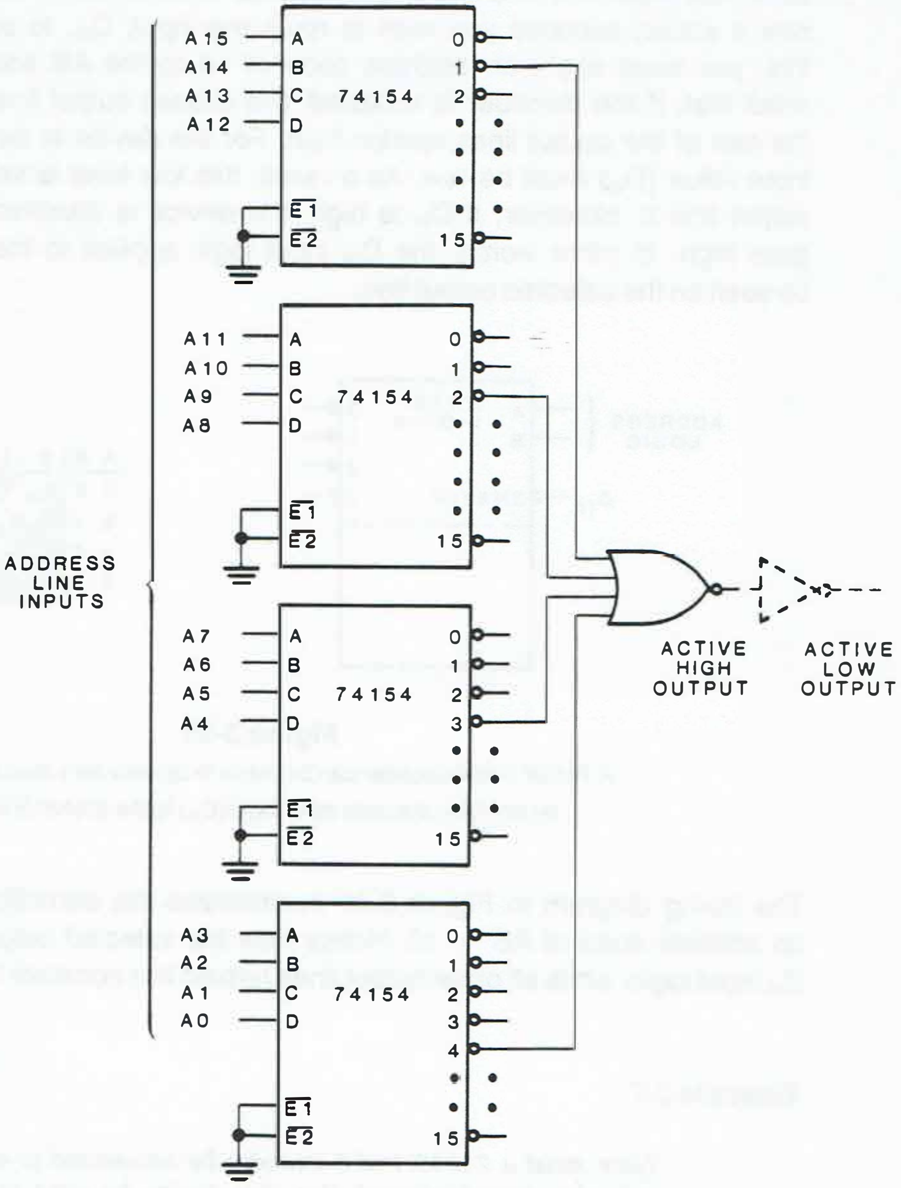


Figure 3-37

A 16-bit address bus can be completely decoded using four 74154 MSI devices.

One section of a 74139 dual 1-of-4 decoder/demultiplexer is shown in Figure 3-39. To make it operate as a demultiplexer, you simply use the ENABLE line as a data input line and apply the address code to the A and B inputs. Here's how it works: suppose you wish to route the input, D_{in}, to output line 2. To do this, you must apply an address code of 10 to the AB address inputs. Now, recall that, **if the decoder is enabled**, this causes output line 2 to go low, while the rest of the output lines remain high. For the device to be enabled, the data input value (D_{in}) must be low. As a result, this low level is seen on the selected output line 2. However, if D_{in} is high, the device is disabled and output line 2 goes high. In other words, the D_{in} input logic applied to the ENABLE line will be seen on the selected output line.

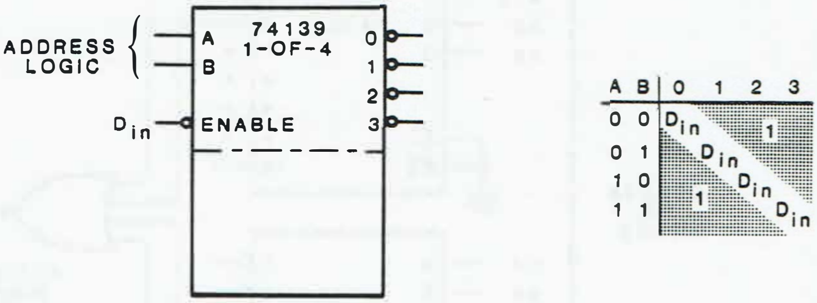


Figure 3-39

A 74139 1-of-4 decoder can be made to operate as a demultiplexer by applying the data input logic (D_{in}) to the ENABLE line.

The timing diagram in Figure 3-40 summarizes the demultiplexer operation for an address code of AB = 10. Notice how the selected output line "follows" the D_{in} input logic, while all other output lines remain in a constant high state.

Example 3-7

How must a 74138 1-of-8 decoder be connected to operate as a 1-of-8 demultiplexer? Can this device be used to generate an inverted output?

A 74138 1-of-8 decoder/demultiplexer is shown again in Figure 3-41. To operate as a demultiplexer, you must connect the D_{in} input logic to one of its three ENABLE lines. The question is, which one? As you can see, I have connected it to the $\overline{E1}$ ENABLE line. As a result, the D_{in} input logic will appear on the output line selected by the address code applied to the A, B, and C input lines. Notice, however, that the $\overline{E2}$ and E3 ENABLE lines must be hardwired to their respective active states.

Actually, you could also apply D_{in} to the $\overline{E2}$ ENABLE line and accomplish the same thing. In this case, $\overline{E1}$ must be hardwired to ground.

To generate an inverted output, you simply apply the Din input logic to the E3 ENABLE line and hardwire both $\overline{E1}$ and $\overline{E2}$ to ground. When this is done, the selected output line will generate an opposite, or inverted, signal as compared to the D_{in} input line.

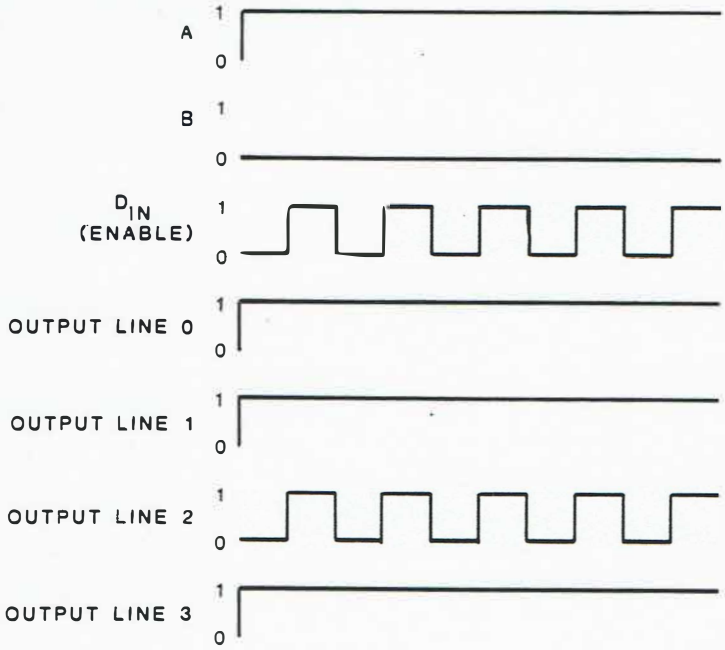
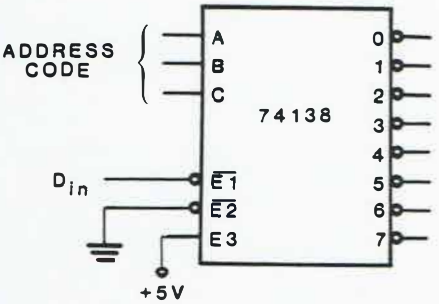
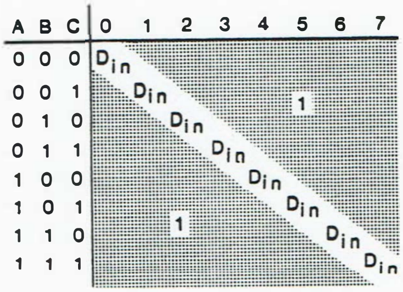


Figure 3-40

A timing diagram shows how the selected output line of a demultiplexer follows the D_{in} logic. All other output remain high.





A	B	C	0	1	2	3	4	5	6	7
0	0	0	D_{in}							
0	0	1		D_{in}				1		
0	1	0			D_{in}					
0	1	1				D_{in}				
1	0	0					D_{in}			
1	0	1			1			D_{in}		
1	1	0							D_{in}	
1	1	1								D_{in}

Figure 3-41

A 74138 1-of-8 demultiplexer.

Self-Test Review

13. Another name for a 4-line-to-16-line decoder is a ____________.

14. Construct a 1-of-8 decoder from 7442 1-of-10 decoder.

15. Design a decoder circuit to implement the two-of-four design. Recall that this design must detect when any two-of-four inputs are high.

16. How many of what type ICs are required to implement the design in problem 15? ____________________

17. How does the cost of implementing the decoder design in question 15 compare to the multiplexer designs in the last section? __________
____

18. The three functional buses of a computer system are __________, __________, and __________.

19. Design an address decoder to completely decode decimal address 8000.

20. How must a 74154 decoder be connected to operate as a 1-of-16 demultiplexer? ______________________________
____
____

21. Can a 7442 decoder be connected to operate as a 1-of-8 demultiplexer? If so, how? ______________________________
____
____
____

Answers

13. 1-of-16 or binary-to-hex decoder

14. A 7442 1-of-10 decoder can be used as a 1-of-8 decoder by using only the three least significant input lines, B, C, and D, and the first eight output lines, 0 through 7.

15. See Figure 3-42 on Page 3-64.

16. A 74154 1-of-16 decoder IC and a 7430 8-bit NAND IC.

17. As of this writing, the decoder design was more costly to implement than any of the multiplexer designs for this problem.

18. data, address, control

19. The hexadecimal equivalent of decimal address 8000 is 1F40. The design requires four 1-of-16 decoders driving a 4-bit NOR gate as shown in Figure 3-43 on Page 3-64. The NOR gate output will go high when this address appears and low for all other addresses.

20. You must apply the D_{in} input logic to one of the two 74154 ENABLE lines. The other line must be hardwired to ground. The D_{in} input logic will then appear on the output line selected by the A, B, C, and D input lines.

21. Yes, you must apply the D_{in} input logic to the most significant input line, A, then use the three least significant lines, B, C, and D, to select 1-of-8 output channels. Normally, you would apply the D_{in} logic to an enable line, but notice that the 7442 does not have any enable lines.

Before you read the next section, go to Unit 8 and perform Experiments 8 and 9.

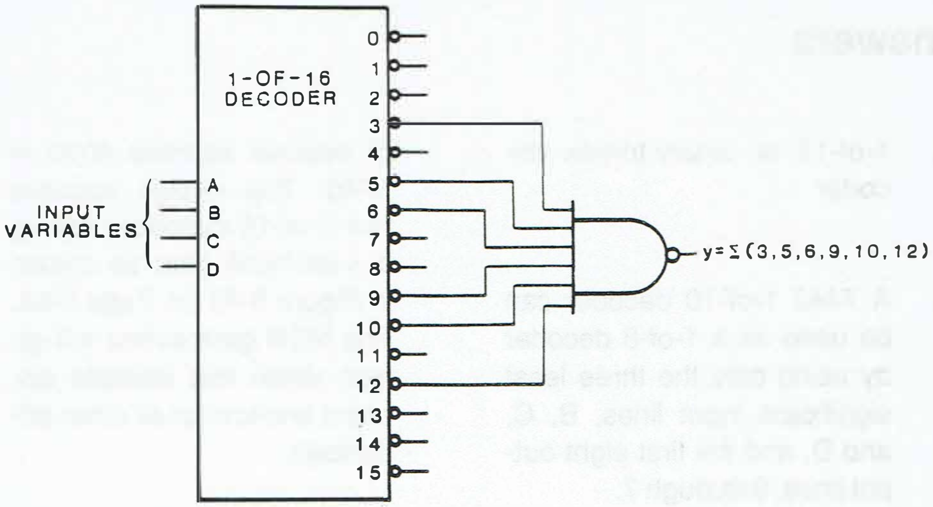


Figure 3-42

A decoder design for the 2-of-4 detector problem.

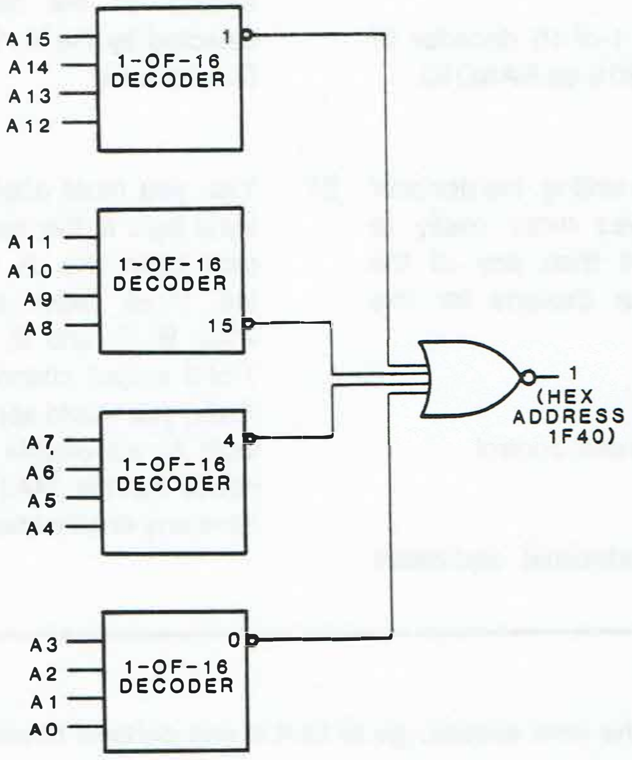


Figure 3-43

An address decoder for hex address 1F40.

READ ONLY MEMORIES (ROMs)

The simplest way to implement combinational logic is to use a read only memory, or ROM. As you will soon discover, a ROM simply stores the logic truth table. Consequently, you can go directly from the truth table to ROM implementation, and avoid logic equations, simplification, and all the design tricks associated with SSI and MSI implementation. A ROM can implement a truth table of several input and output variables directly, without any external SSI or MSI logic.

So, your probably wondering: "Why should I even consider an SSI or MSI design?" Why not use ROMs to implement all combinational logic? It boils down to a question of economics. A ROM design is only cost effective when there are several input and output variables. In fact, a good rule of thumb is to consider a ROM design only when there are at least four or more input and output variables, and no standard MSI devices available to implement the design. Thus, for complex designs involving four or more input and output variables, ROM implementation becomes the most efficient and cost effective solution.

A Simple Diode ROM

A ROM made from diodes is shown in Figure 3-44 on Page 3-66. Observe that it is basically an m × n conductor matrix containing m rows and n columns. The rows form eight memory **registers** labeled R0 through R7, and the columns, labeled D0 through D3, generate the bits within each register. Since the ROM shown in Figure 3-44 contains eight 4-bit registers, it is called an 8 × 4 ROM.

Diode connections between the rows and columns create **cells** that produce either a logic 0 or logic 1. Here's how it works. Suppose the switch in Figure 3-44 on Page 3-65 is pointing to register R0 as shown. This applies a ground potential, or TTL logic 0, to row 0 of the conductor matrix. Observe that there is a diode connection between this row and column D3, but no connection between this row and column D2. Electrons flow up through the switch, through the forward biased diode, and through the D3 pull-up resistor to the +5 volt supply. Ideally, the D3 pull-up resistor drops the entire five volts, leaving a ground, or logic 0, potential at point D3.

On the other hand, electrons cannot flow through the D2 resistor because there is no diode connection between the R0 row and D2 column. Thus, the D2 resistor doesn't drop any voltage and the entire +5 volt, or logic 1, potential is seen at point D2. You might think that the diodes in the matrix could be replaced by jumper wires. However, this won't work since the registers could not be electrically isolated from each other.

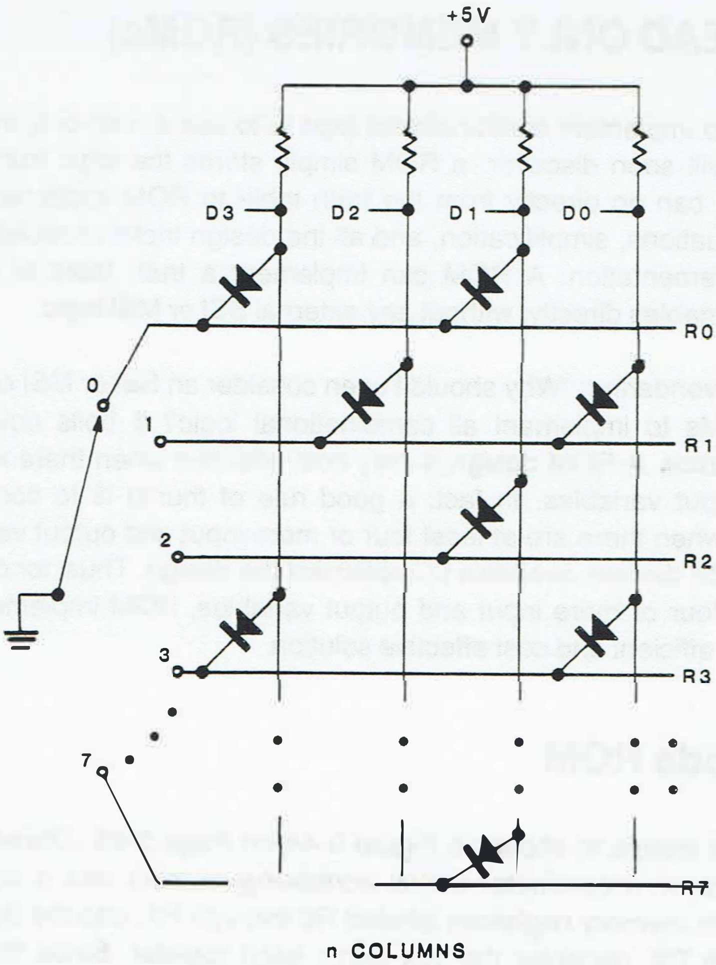


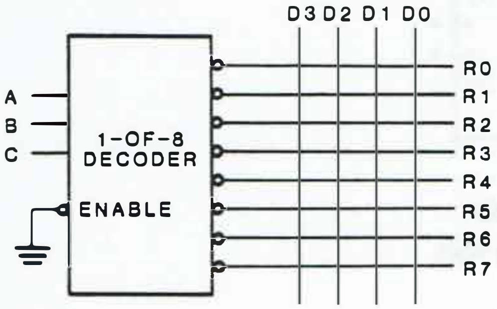
Figure 3-44

An m × n diode ROM.

To summarize, a given matrix cell produces a logic 0 if it contains a forward biased diode between its row and column, otherwise it produces a logic 1. Using this idea, the value "stored" in register R0 is 0101. Likewise, the value stored in register R1 is 1010, and so on. The ROM in Figure 3-44 is called a diode ROM, for obvious reasons.

The seven switch positions in Figure 3-44 are called the **register addresses**, since they determine which particular register is selected. Switch position 0 is the address of register R0, position 1 is the address of R1, and so forth. By changing the switch position to different register addresses, you are actually **distributing** a ground, or logic 0, potential to the various registers.

What type of MSI device are you aware of that performs this distribution function? How about a decoder/demultiplexer? Suppose that you replace the switch in Figure 3-44 with a 1-of-8 decoder/demultiplexer as shown in Figure 3-45. Now, rather than a switch position, the registers are selected by the logic applied to the decoder input lines, ABC. The result is the truth table shown on the right side of the figure. Notice that an address of ABC = 000 selects register R0, address 001 selects R1, and so on. In each case, the respective register contents are seen at points D3, D2, D1, and D0.



A	B	C	D3 D2 D1 D0
0	0	0	R0
0	0	1	R1
0	1	0	R2
0	1	1	R3
1	0	0	R4
1	0	1	R5
1	1	0	R6
1	1	1	R7

Figure 3-45

The use of a decoder minimizes the number of address lines required for a ROM.

Notice what the ROM address decoder does for you. Suppose you wanted to construct a 1K × 8 ROM. Such a ROM would have 1024 8-bit registers. Without a decoder you would need a switch with 1024 different positions, one for each register address. It is highly unlikely that you could find such a switch. However, a decoder circuit would only require ten address lines to select the same 1024 registers, since $2^{10} = 1024$. Granted, a commercially available 1-of-1024 decoder is not available, but such a decoder can be easily integrated into an LSI ROM device.

The diode ROM in Figure 3-44 provides a good model and teaching tool to illustrate the operation of a ROM. In fact, you can build such a ROM using an MSI decoder and discrete diodes. However, single diodes are not actually integrated into commercial ROM devices. Instead, bipolar or MOS transistors are integrated into the ROM cells. The junctions of the integrated transistors are then used to form the diodes within the matrix.

The ROM Symbol and Three State Buffering

A black-box symbol for a 1k × 8 ROM is shown in Figure 3-46. Here, the ten decoder address lines are labeled A0 through A9, and the eight data output lines are labeled D0 through D7.

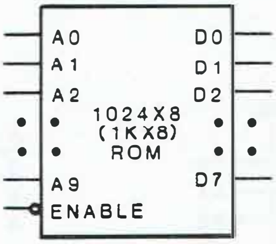


Figure 3-46
A typical ROM symbol.

You will also note that the ROM symbol includes an ENABLE line. Again, the ENABLE line acts as an on/off switch for the ROM, but it is different from the enabling function of MSI devices. Recall that when a device is enabled, it is switched on and its outputs generate logic according to its respective function. The same is true of a ROM. However, when a multiplexer or decoder device is disabled, its outputs are held in a constant logic 0 or logic 1 state. This is **not** true of a ROM and most other LSI devices. When a ROM is disabled, its output lines do not produce any logic. Rather, they are put in an **open** circuit, or **high impedance**, state. Its as if the ROM output lines were totally disconnected from the circuit.

This idea is called **three state,** or **tri-state buffering**. The term "three state" is used since the output lines can take-on three possible states: logic 0, logic 1, or open circuit. The 0 and 1 states are produced as a result of the ROM operation when it is enabled. The open circuit state results when the ROM is disabled.

Three state buffering is extremely important in computer bus systems, since it allows the system to connect and disconnect various devices to and from the bus as they are needed during the execution of a program. More about this later.

Example 3-8

A ROM chip has 12 address lines and 8 data lines. How many and what size registers does it contain? How would this ROM be listed in a manufacturer's catalog?

Since the ROM chip has 12 address lines, it must have 2^{12}, or 4096, internal registers. Since it has eight data lines, there must be eight storage bits in each register. As a result, the ROM would be listed as a 4096 × 8 ROM in a manufacturer's catalog. This ROM might also be listed as a 4K × 8 or 32K bit ROM. Of course, K = 1024. Notice that the 32K bit designation does not tell you the internal organization of the device, only that it contains 32K 1-bit storage cells.

ROM Devices

There are a variety of read only memory devices available today. They include the basic **mask-programmed ROMs** (ROMs), **one-shot programmable ROMs** (PROMs), **erasable programmable ROMs** (EPROMs), and **electrically erasable ROMs** (EEROMs). Each of these devices has its particular advantages and disadvantages. As each type of device is discussed, keep in mind that the application will clearly determine the type of ROM to be used.

MASK-PROGRAMMED ROM

A mask-programmed ROM is always programmed by the ROM manufacturer. This type of ROM is available with "canned" programs such as system monitors, editors, interpreters, etc. Also, many of the personal computer business and game packages are available in mask-programmed ROM. When this is the case, the ROM is usually encased in a plug-in cartridge. You simply plug the cartridge into a compatible system; then you can begin using the canned program immediately.

To implement a digital design with a ROM, you will need to specify the truth table to be mask-programmed into the ROM. The truth table data then becomes the program to be entered into the ROM device. This is called **custom programming.** The Rom device data sheet will specify how your truth table data is to be formatted and forwarded to the manufacturer. Most manufacturers require that the information to be programmed must be forwarded to them on punched paper tape, IBM punch cards, EPROM, or floppy disk. However, you must order many hundreds or thousands of units to justify the cost.

In addition to the cost of each ROM device, the manufacturer will charge you with a one-time mask charge. The mask charge is normally several thousand dollars. Thus, the more ROMs you order, the cheaper each individual ROM becomes, since the mask charge is amortized over more devices. Compared to other ROM devices, mask-programmed ROM is the most economical as long as you order several hundreds or thousands of devices.

The biggest disadvantage of mask-programmed ROM is that once it is programmed, the program is fixed and cannot be changed. Any alteration in the program will require a new program mask, and thus generate a new mask charge.

To program the ROM, the manufacturer makes a **metallization mask** from your truth table data. The metallization mask is used in the last step of the chip fabrication process to define the 1's and 0's in the ROM memory cells. Two cells of a mask programmed ROM are shown in Figure 3-47. The ROM cells are defined by rows and columns like the diode ROM discussed earlier. Each cell of a MOS ROM contains an integrated MOSFET device. The metallization mask will cause the gate lead of the MOSFET to be connected for a logic 1 or not connected for a logic 0. When the gate lead is connected, the MOSFET can be turned on by applying a potential to the respective row line, which results in a logic 1 level on the corresponding column line. If the gate lead is not connected, the MOSFET cannot be turned on and no potential, or logic 0, appears on the corresponding column line.

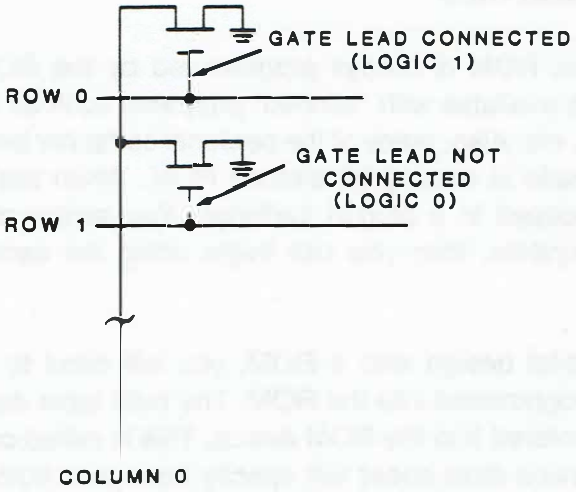


Figure 3-47

Two cells of a MOS mask-programmed ROM.

The ROM cells can contain MOSFET devices (shown in Figure 3-47) or bipolar devices. The advantage of a bipolar ROM is speed. Bipolar ROMs will have access times anywhere between 50 nanoseconds and 150 nanoseconds. On the other hand, MOS ROMs will provide access times between 200 nanoseconds and 400 nanoseconds. However, bipolar devices are generally more expensive, less dense, and require more power to operate.

PROGRAMMABLE ROM (PROM)

A PROM is **user programmable**, meaning that you program the device, rather than the manufacturer. You are not faced with a mask charge, but the cost on a per unit basis is relatively high compared to a large mask-programmed ROM purchase. In addition, you must have a PROM programming unit, or **PROM burner**, to program the device. The PROM burner is actually a small microcomputer that allows you to enter your data or program in its memory and then transfer that program to the PROM. The PROM burner also supplies the voltage and current levels required to program the device.

Two cells of a PROM are shown in Figure 3-48. Most PROMs are bipolar devices and, therefore, each memory cell contains an integrated bipolar transistor. The transistor emitter is connected to the cell column line through a **fuse link**. When the PROM is manufactured, all of the fuse links are connected, and thus all cells contain logic 1's. To program the device, the PROM burner programs 0's into the required cells by "blowing" the fuse links. This is accomplished by applying a current that exceeds the current-carrying capability of the fuse link. To open the fuse link, a typical PROM will require a programming voltage of 8 to 12 volts and a programming supply current of 400 to 600 milliamps.

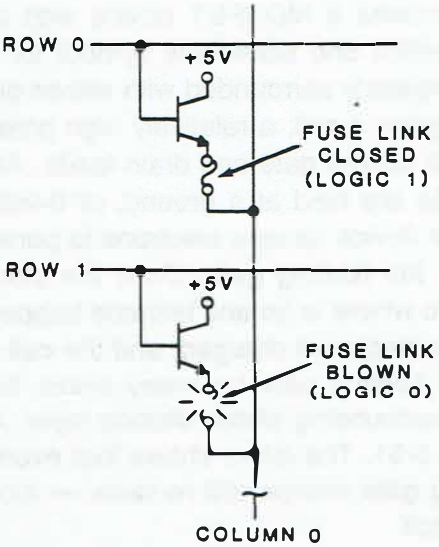


Figure 3-48

Two cells of a bipolar PROM.

Once the PROM is programmed, it cannot be reprogrammed; if changes must be made to the program, the old PROM must be discarded and a new one programmed. For this reason, PROMs are sometimes referred to as **one-shot ROMs**.

ERASABLE/PROGRAMMABLE ROM (EPROM)

The EPROM is user-programmable **and** erasable. Unlike a PROM, an EPROM can be reprogrammed many times. For this reason, EPROMs are becoming much more popular than PROMs.

An EPROM is pictured in Figure 3-49. It is easy to recognize by the quartz window above its integrated circuit, or chip. The quartz window is provided so that the chip can be exposed to ultraviolet light for erasure. Exposing the window to a high intensity ultraviolet light source for approximately 30 minutes is all that is required to erase any information stored in the device.

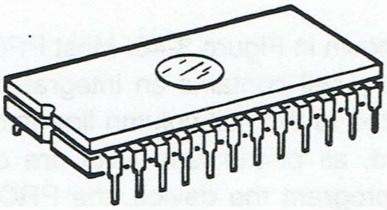


Figure 3-49

A typical EPROM device.

Each EPROM memory cell contains a MOSFET device with a **floating gate**. Figure 3-50 shows a cross section and schematic symbol for one cell of this device. The floating gate is completely surrounded with silicon dioxide, an excellent insulating material. To program a cell, a relatively high potential of approximately 25 volts is applied to the control gate and drain leads. At the same time, the source and P-type substrate are held at a ground, or 0-volt, potential. The difference in potential within the device causes electrons to penetrate the silicon dioxide insulating layer around the floating gate. Once the electrons penetrate the insulating layer, they have no where to go and become trapped by the floating gate. The floating gate therefore becomes charged, and the cell is programmed. The charge will remain on the floating gate for many years, because there is very little leakage through the surrounding silicon dioxide layer. A typical charge decay curve is shown in Figure 3-51. The curve shows that even after a 10-year period, over 80% of the floating gate charge still remains — more than enough to maintain a logic 1 level in the cell.

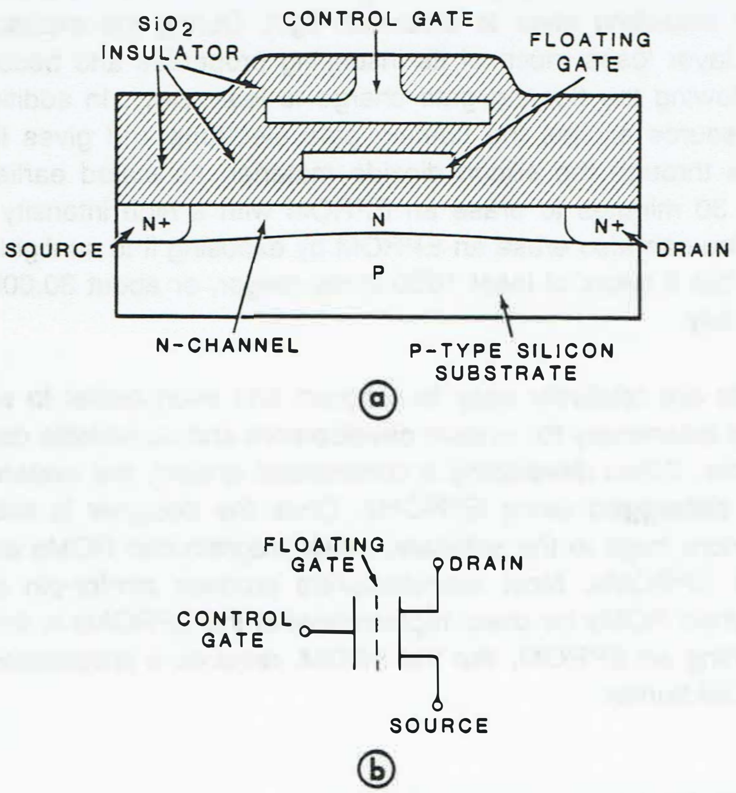


Figure 3-50

EPROM memory cell cross-section (a) and schematic symbol (b).

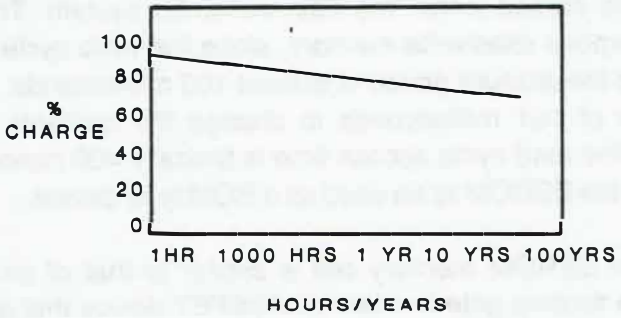


Figure 3-51

Charge decay of a typical EPROM.

You can eliminate the floating gate charge and erase the cell by exposing the silicon dioxide insulating layer to ultraviolet light. During the exposure period, the insulating layer loses much of its insulating properties and becomes more conductive, allowing the floating gate charge to leak away. In addition, the ultraviolet light source excites the floating gate electrons and gives them more energy to leak through the silicon dioxide material. As stated earlier, it takes approximately 30 minutes to erase an EPROM with a high intensity ultraviolet light source. You can also erase an EPROM by exposing it to sunlight or normal room lighting, but it takes at least 1000 times longer, or about 30,000 minutes, to erase it properly.

Since EPROMs are relatively easy to program and even easier to erase, they are being used extensively for system development and nonvolatile data storage in small systems. When developing a commercial system, the system software is stored and **debugged** using EPROMs. Once the designer is satisfied that there are no more bugs in the software, mask-programmed ROMs are ordered to replace the EPROMs. Most manufacturers produce pin-for-pin compatible mask-programmed ROMs for direct replacement of the EPROMs in their product line. Programming an EPROM, like the PROM, requires a programming device called an EPROM burner.

ELECTRICALLY ERASABLE ROM (EEROM)

The last type of ROM that you need to know about is the electrically erasable ROM, or EEROM. This type of ROM is also called E^2ROM and **electrically alterable ROM**, or **EAROM**. A more generic term for these devices is **read-mostly memory**. They can actually be classified as read/write memories, but the write cycle is very long compared to the read cycle, thus the term read-mostly memory.

With the proper software, hardware gating, and timing signals, EEROMs can be programmed and erased within the host computer system. They cannot be used as general purpose read/write memory, since the write cycle takes at least one millisecond and the erasure period is at least 100 milliseconds. Thus, it would require a minimum of 101 milliseconds to change the contents of a memory location. However, the read cycle access time is typically 400 nanoseconds. This is sufficient to allow the EEROM to be used as a ROM type device.

The structure of the EEROM memory cell is similar to that of an EPROM cell. Each cell contains a floating gate N-channel MOSFET device that is programmed by storing a charge on the floating gate. The cell is erased by applying a strong current pulse to neutralize the stored charge.

The advantage of using EEROMs is that they do not have to be removed from the system to be programmed or erased. However, they have several disadvantages. Most EEROM devices are relatively low density devices. They are expensive as compared to other ROM devices. They are not very reliable after many erasures. Finally, they cannot retain data as long as EPROM devices. For these reasons, the EEROM is not as widely used as the EPROM. But as processing technology improves, EEROM devices will probably be used in more industrial and commercial microcomputer systems.

A comparison summary of the different ROM devices just described is provided in Figure 3-52.

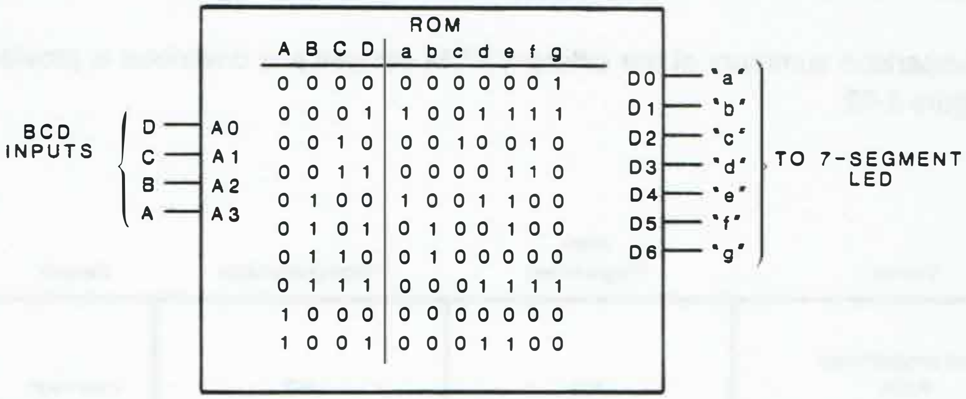
Device	User-Programmed	Reprogrammable	Density	Cost
Mask-programmed ROM	NO	NO	Very High	Lowest in large quantities
PROM	YES	NO	Low	Medium
EPROM	YES	YES	High	Medium
EEROM	YES	YES	Low	High

Figure 3-52
ROM comparisons.

Implementing Combinational Logic Using ROMs

You can think of a ROM as simply a big look-up table. To imagine this, think of a typical code conversion problem. One simple problem that you are familiar with is conversion from BCD to 7-segment code. Recall that the initial black-box concept for this problem requires four BCD inputs, a black-box that performs the code conversion, and seven outputs driving the LED.

The required black-box is shown again in Figure 3-53. Notice that the truth table for a common anode LED is shown inside the box. This was done in order to illustrate the idea that the black-box circuit simply performs a look-up function via the truth table. You apply the BCD code to the inputs and the circuit generates the corresponding output code according to the programmed truth table. This is the function of any digital design, to implement the truth table.



A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0

Figure 3-53

The black-box concept and ROM implementation for a 7-segment common anode LED.

Now, suppose the black-box is simply the ROM in Figure 3-53 that has been programmed with the 7-segment LED truth table data. The address lines of the ROM are the BCD input lines and the ROM data lines are the 7-segment output lines. The ROM cells contain the logic 1's and 0's from the truth table. If this is the case, an address input of A3 A2 A1 A0 = 1 0 0 0 causes register R8 of the ROM to be selected. The contents of this register, 00000000, appear on the ROM data lines which causes the common anode LED to display the value 8. That's all there is to it!

In summary, when using a ROM to implement a logic design, the ROM address lines become the circuit input lines and the ROM data lines are the circuit outputs. Consequently, the number of ROM address lines determines the maximum number of input variables that can be implemented with a given ROM. Likewise, the number of ROM data lines determines the maximum number of logic equations that can be implemented with a given ROM. Of course, the ROM must be programmed with the truth table logic for each output function.

Table 3-4 summarizes this idea by listing the maximum logic implementation capabilities for several common ROM packages.

Table 3-4: Maximum Logic Implementation Capabilities of Several Common ROM Devices

ROM Device	Max. Number Input Variables	Max. Number Logic Functions
32 × 8	5	8
256 × 4	8	4
256 × 8	8	8
512 × 4	9	4
512 × 8	9	8
1024 × 4	10	4
1024 × 8	10	8
2048 × 4	11	4
2048 × 8	11	8

Example 3-9

Design a digital circuit that will multiply any two 2-bit binary numbers. Use a ROM to implement the design.

The first step is to develop a black-box concept as shown in Figure 3-54. The inputs are two 2-bit numbers: AB and CD. The output lines, D0 through D3, must generate the binary logic required to represent the product. The maximum product of two 2-bit numbers is 1001. Thus, four output lines are required.

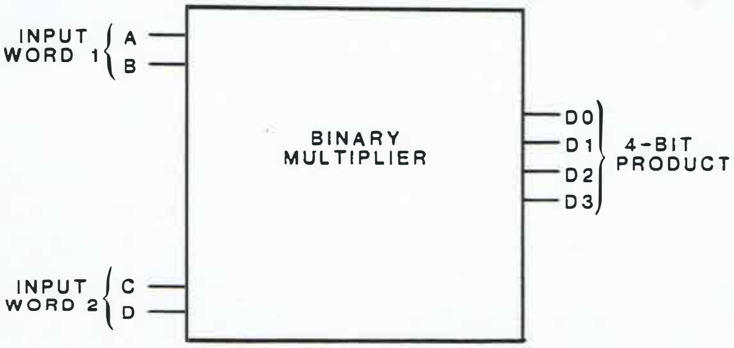


Figure 3-54

The black-box concept for a binary multiplier.

In a ROM design, the ROM **is** the black-box. To implement the design, the truth table must be programmed into the ROM. Once this is done, the ROM address lines are connected to the circuit inputs and the ROM data lines generate the circuit outputs. The completed design for this problem is shown in Figure 3-55. This relatively simple design problem only requires a $2^4 \times 4$, or 16×4 ROM. Typically, ROMs this small are not readily available. That is why a 32 × 8 PROM is used in the design. Here, the most significant address line, A4, is hardwired to ground and the upper four data lines, D4 through D7, are not used.

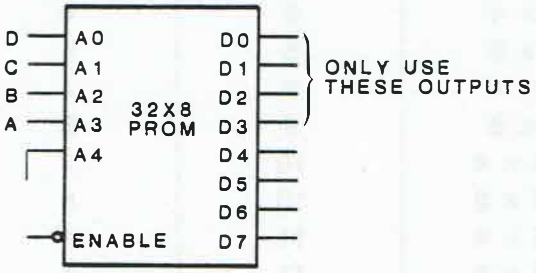


Figure 3-55

A PROM implementation of the binary multiplier (Example 3-9).

This example clearly illustrates the use of a ROM for implementing arithmetic logic. Standard SSI and MSI arithmetic circuits get very complex, especially for multiplication, division, square root, and trig functions. However, when you think about it, all of these arithmetic operations can be implemented with look-up tables. For instance, the appendices of most technical math books are filled with page after page of trigonometric look-up tables. These look-up table type of applications are prime candidates for ROM implementation, since the respective tables can be easily programmed into a ROM.

Self-Test Review

22. A ROM chip has 9 address lines and 4 data lines. What is its internal register organization? ______________.

23. The 2716 is a popular EPROM chip. A manufacturer's catalog indicates that the 2716 is a 2K × 8 EPROM. How many address and data pins would you expect to find on the chip? Address _________, Data _____________.

24. How many addresses must be used to access all the registers in a 4K × 8 ROM? _____________

25. What is the highest register designation, or address, for a 256 × 4 ROM? _______________

26. The feature of a ROM that allows it to be effectively disconnected from a system is called _________________ __________.

27. What is the major advantage and disadvantage of mask-programmed ROM?
Advantage ____
Disadvantage ____

28. How can you tell the difference between a PROM and an EPROM?
____
____

29. Explain how an EPROM stores data in its memory cells. _________
____
____
____

30. Another popular 4-bit binary code used in digital circuits is the **Excess-3**, or **XS3**, code. This code was developed to simplify arithmetic operations with BCD numbers. To determine the code, add 3 to a decimal value before converting it to binary. For instance, the XS3 code for the number 5 is the 4-bit binary equivalent of $5 + 3 = 8$, or 1000.

    Design a digital circuit to convert from straight binary to XS3 for the decimal values 0 through 9.

Answers

22. 512 × 4

23. Address 11 — $2^{11} = 2K$.
Data 8 — there are eight bits per register.

24. 4096

25. 255

26. three-state buffering

27. Advantage — cost, when large quantities are required. Disadvantage — must be programmed by the ROM manufacturer. Any changes require a new mask.

28. The EPROM has a transparent quartz window above its integrated circuit, the PROM does not.

29. The cells are programmed by applying a +25-volt potential difference between the MOSFET control gate and source.

30. Solution follows.

Solution to Question 30:

The first thing you must do is develop a table of XS3 versus binary values for the numbers 0 through 9. Using the conversion process explained in the problem, the required table is:

Decimal	Binary ABCD	XS3 D3 D2 D1 D0
0	0000	0 0 1 1
1	0001	0 1 0 0
2	0010	0 1 0 1
3	0011	0 1 1 0
4	0100	0 1 1 1
5	0101	1 0 0 0
6	0110	1 0 0 1
7	0111	1 0 1 0
8	1000	1 0 1 1
9	1001	1 1 0 0

The above table can be most easily implemented using a ROM since there are four binary input bits, ABCD, and four XS3 output bits, D3 D2 D1 D0. To implement the design, the above table must be programmed into a 16 × 4 ROM. Once this is done, the binary inputs are connected to the ROM address lines and the ROM data lines will generate the XS3 code. The completed design is shown in Figure 3-56.

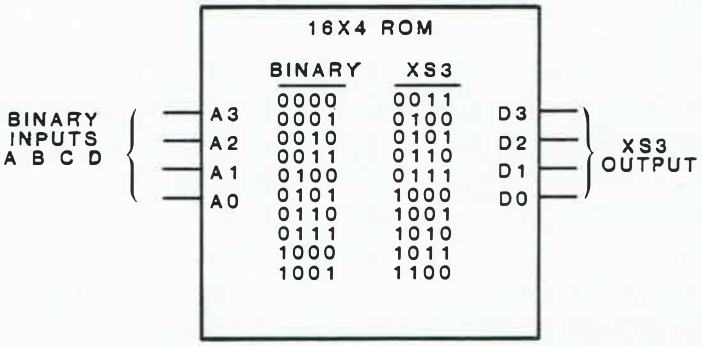


Figure 3-56

A 16 × 4 ROM implementation of the binary to XS3 conversion required by problem 30.

PROGRAMMABLE LOGIC ARRAYS (PLAs)

Programmable logic arrays, or PLAs, are sometimes called **programmable logic devices**, or **PLDs**. Like a ROM, this LSI device can be programmed to implement just about any logic function. However, unlike a ROM, programmable logic arrays do not contain programmable memory cells. Instead, a PLA contains an array of AND/OR gates that can be programmed to generate any sum-of-product logic equation.

A PLA can be programmed to implement as many as 10 logic equations with up to 16 input variables. As a result, PLAs are commonly used to implement very complex logic functions, where SSI and MSI devices result in too many ICs and ROM implementation is not cost effective.

The Internal Structure of a PLA

A programmable logic array is exactly what its name implies: an array, or matrix, of AND/OR logic gates that can be interconnected via a program mask to generate any logic function. Recall from the SSI design process, that you always end-up with a sum-of-products equation that results in an AND/OR design. Now, suppose that you had a breadboard full of unconnected AND/OR gates. Theoretically, if you had enough gates you could follow any Boolean expression and make connections to the gates to implement any logic operation. The idea is the same with a PLA. However, rather than being on a breadboard, the AND/OR gates are integrated into an LSI device called a PLA. Also, rather than making manual connections between the gates, the gate connections are made via a program generated mask that is derived directly from your logic equation(s).

A small portion of a PLA is shown in Figure 3-57. As you can see, it is basically an AND/OR matrix structure. The AND gates form the rows and the OR gates form the columns of the matrix. As a result, the rows are commonly called the **AND plane** and the columns are called the **OR plane**. Recall that any SOP expression is simply a series of AND terms grouped together by an ORing operation. This is exactly the idea behind a PLA since its structure, from input to output, is simply a series of AND gates followed by OR gates.

When using a PLA, the design variables are applied to the PLA inputs as shown. Notice that each variable is then available as a non-inverted or inverted input to the AND gates via a set of bus lines. This allows any of the input variables or its complement to be applied to one or more of the AND gates. These AND gates generate the product terms of the input variables. The product terms formed

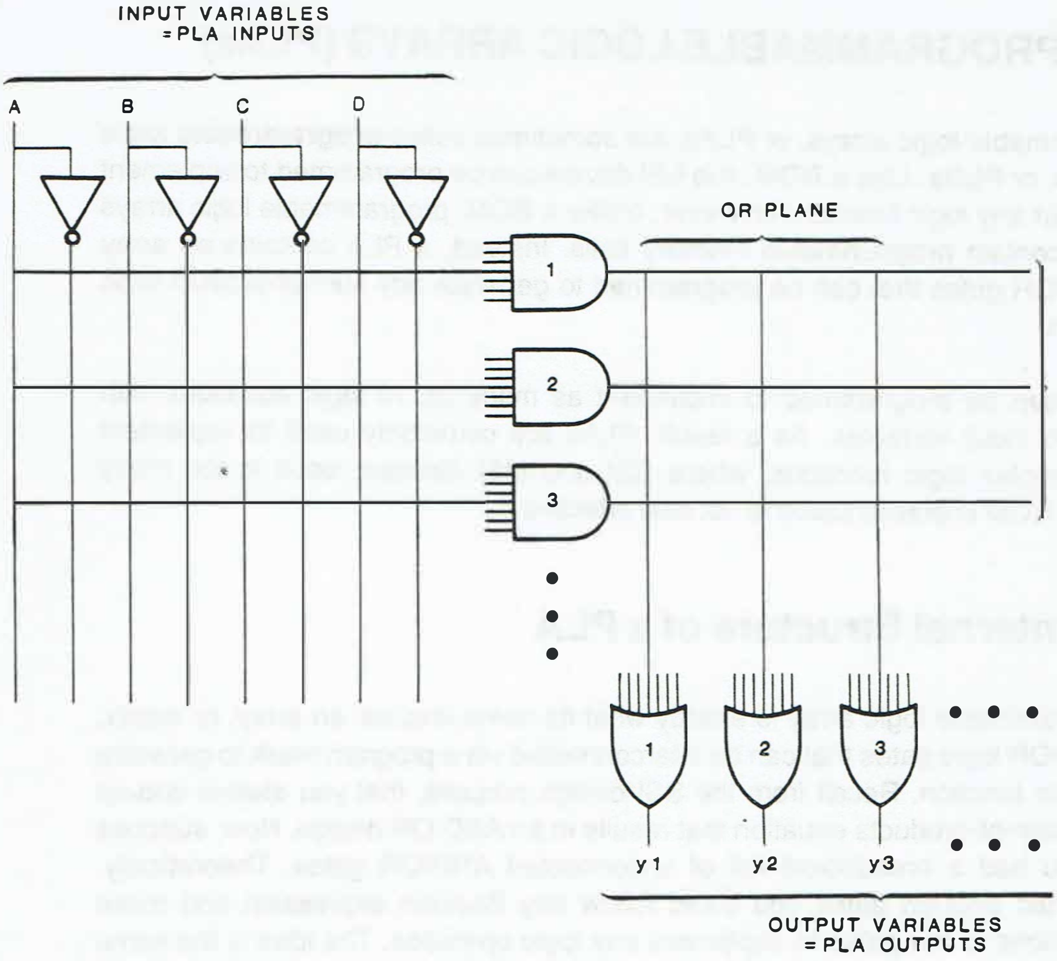


Figure 3-57
A small section of a PLA.

by the AND gates are then summed via the OR gates to form the output functions (y_1, y_2, etc.). The result is a sum-of-product output.

The selection of which input variables are to be connected to the AND gates, and which AND gate outputs are applied to a given OR gate is determined when the device is programmed. Of course, the program is derived directly from your truth table or logic equations. With a standard PLA, the interconnections are masked during the last stage of the manufacturing process, like a mask-programmed ROM. Field programmable devices, or **FPLAs**, can be programmed using a computerized programming device similar to a PROM burner. Both standard PLAs and FPLAs employ bipoler technology.

Example 3-10

Using the PLA structure shown in Figure 3-58, write the SOP logic equation generated at the y_1 output. Assume that the PLA has been programmed to make a connection at each point you see an asterisk (*).

Following the y_1 output back, you see that AND gates 1, 3, and 5 are driving OR gate 1. Thus, the SOP expression has three terms, one for each AND gate. Following the AND gate inputs back to the input bus lines you get the following terms:

AND gate 1: $A\overline{B}\overline{C}$
AND gate 3: $\overline{A}BC$
AND gate 5: $\overline{A}\overline{B}C$

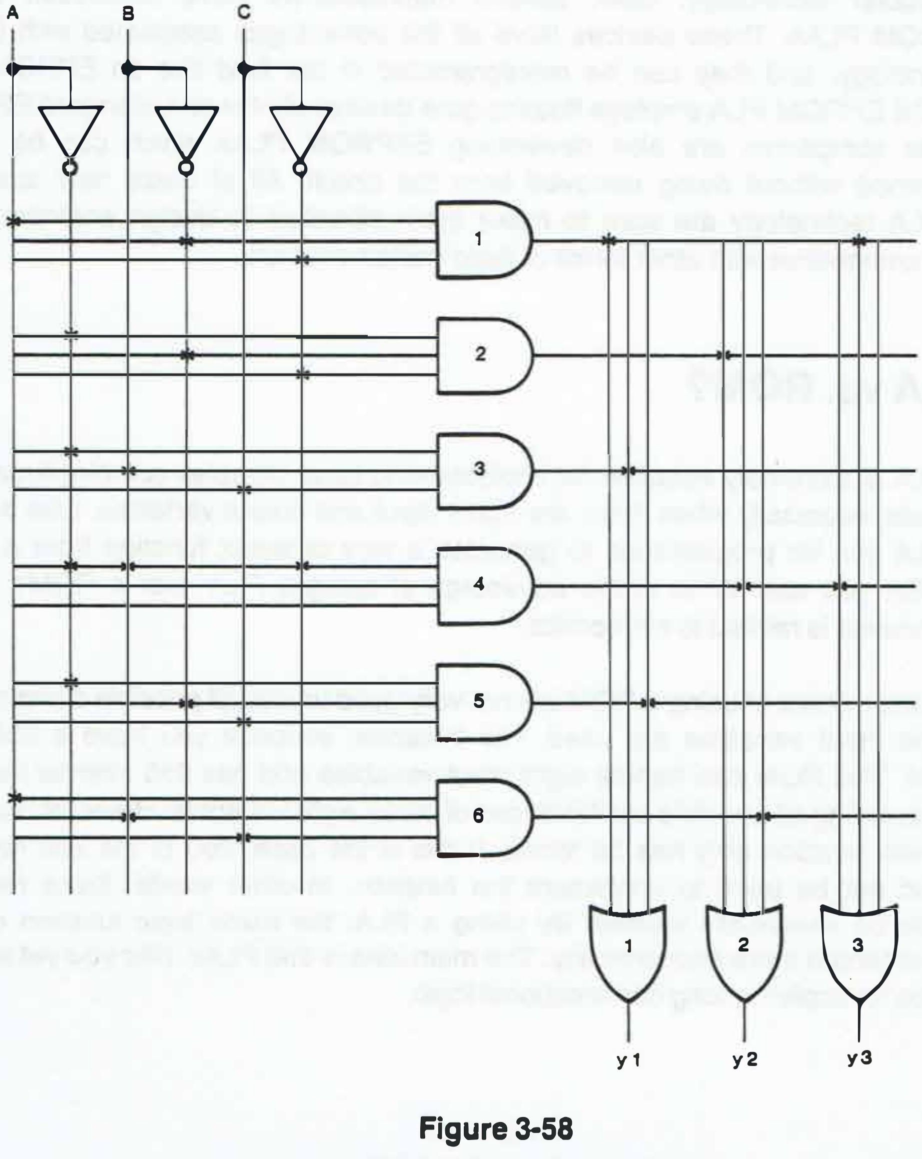


Figure 3-58

A programmed PLA for Example 3-10.

Finally, ORing the above terms together gives you:

$$y_1 = A\overline{B}\,\overline{C} + \overline{A}BC + \overline{A}\,\overline{B}C$$

PLA Packages

Typical PLA chip sizes range from 20 to 24 pins per device. Such a device will contain from 200 to 300 gates in the array. Special chip packages using pin grids are available with chip densities exceeding 1000 gates. Most PLAs are TTL devices; however, ECL and CMOS devices are also available.

You should be aware that PLA technology is improving and finding increased applications in the industrial environment. Until recently, PLAs were only available in bipolar technology. Now, several manufacturers have introduced CMOS EPROM PLAs. These devices have all the advantages associated with CMOS technology, and they can be reprogrammed in the field like an EPROM. The CMOS EPROM PLA employs floating gate devices similar to a standard EPROM. Some companies are also developing EEPROM PLAs which can be reprogrammed without being removed from the circuit. All of these new advances in PLA technology are sure to make them attractive to design engineers, and cost competitive with other forms of logic implementation.

PLA vs. ROM?

A PLA is extremely valuable for implementing large complex combinational logic circuits, especially when there are many input and output variables. Like a ROM, a PLA can be programmed to generate a very complex function from a single IC. So, you ask: What is the advantage of using a PLA over a ROM? Again, the answer is related to economics.

The economics of using a ROM are not very good unless all possible combinations of the input variables are used. For instance, suppose you have a 256 × 8 ROM. This ROM can handle eight input variables and has 256 internal registers representing all possible combinations of these eight variables. However, suppose a given function only has 56 terms. If this is the case, 200 of the 256 registers would not be used to implement the function. In other words, these registers would be essentially wasted. By using a PLA, the same logic function can be implemented more economically. The main idea is that PLAs offer you yet another option for implementing combinational logic.

Self-Test Review

31. What form of logic expressions are implemented directly with a PLA?

____

32. What type of gates are integrated into a PLA? ______________

33. The user programmable version of a PLA is called a ________.

34. Explain what happens within a PLA when it is programmed. ________

____

____

35. Write the SOP expression generated by the y_2 output in Figure 3-58 on Page 3-85.

____

Answers

31. Sum-of-product

32. AND/OR

33. field programmable PLA, or FPLA

34. When a PLA is programmed, gate connections are made to the AND/OR gate matrix to implement an SOP expression. The program generated mask defines which input variables are to be connected to the AND gates, and which AND gate outputs are to be applied to a given OR gate.

35. $y_2 = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}B\overline{C} + ABC$

SUMMARY

Multiplexers and decoders are MSI devices used to implement combinational logic. A multiplexer, or data selector, is nothing more than an electronic switch that is used to select and route one of many inputs to a single output. Multiplexers can be used to implement a single logic equation directly, by applying the truth table logic to the multiplexer inputs. This is the obvious means of implementing combinational logic with a multiplexer.

Smaller, less expensive multiplexers can be used to implement the same logic using special design techniques. These techniques are referred to as the n – 1 and n – 2 multiplexer design solutions, where n is the size of the multiplexer required for the obvious solution. As the names imply, these design techniques reduce the size of the multiplexer by powers of n – 1 and n – 2, respectively.

A decoder is a device that recognizes a given binary input value. Decoders are used to implement combinational logic by connecting the proper decoder output lines to the inputs of a NAND gate. The logic input variables are connected to the decoder input lines. Decoders have an advantage over multiplexers, since a single decoder can be used to implement several logic functions that have the same input variables.

A decoder can also be used as a demultiplexer, or data distributor. A demultiplexer is the opposite of a multiplexer, since it distributes a single input to one of many outputs. Decoder ICs are made to act as demultiplexers by connecting a single data input to one of the decoder enable lines.

Read only memories and programmable logic arrays are LSI devices used to implement combinational logic. These devices are used when at least four logic functions with four or more input variables must be implemented. With a ROM, the logic truth table is actually programmed into the device. Different types of ROMs are available to satisfy different applications. They include mask-programmed ROMs, PROMs, EPROMs, and EEROMs.

A programmable logic array, or PLA, is an LSI device that contains an array of AND/OR gates. A PLA is programmed directly from the SOP logic equation. Internal AND gate connections are made to implement the product terms of the equation, while internal OR gate connections are used to sum the various product terms and generate the SOP expressions. PLAs are extremely economical when you must implement several complex functions containing many input variables.

Before you complete the final examination, go to Unit 8 and perform Experiment 10.

UNIT EXAMINATION

1. A multiplexer is a:
   - A. Decoder.
   - B. Data distributor.
   - C. Data selector.
   - D. None of these.
2. A 100-to-1 multiplexer circuit must have __________ channel select lines.
   - A. Five.
   - B. Six.
   - C. Seven.
   - D. Eight.
3. How many 74153 dual 4-to-1 multiplexer ICs are required to make an 8-to-1 multiplexer?
   - A. One.
   - B. Two.
   - C. Three.
   - D. Four.
4. Using the obvious solution for combinational logic design with multiplexers, a circuit with four input variables requires a:
   - A. 4-to-1 multiplexer.
   - B. 8-to-1 multiplexer.
   - C. 16-to-1 multiplexer.
   - D. 32-to-1 multiplexer.
5. Using the n-1 solution for four input variables requires a:
   - A. 4-to-1 multiplexer.
   - B. 8-to-1 multiplexer.
   - C. 16-to-1 multiplexer.
   - D. 32-to-1 multiplexer.

6. How many input pairs would be generated for three input variables using the n-1 multiplexer design solution?
   - A. Two.
   - B. Four.
   - C. Six.
   - D. Eight.

7. Another name for a 3-line-to-8-line decoder is a:
   - A. 1-of-8 demultiplexer.
   - B. 8-to-1 multiplexer.
   - C. Binary-to-BCD decoder.
   - D. Octal-to-hex decoder.

8. What size decoder is required to implement a design of three input variables?
   - A. 1-of-3.
   - B. 8-to-1.
   - C. 1-of-8.
   - D. 1-of-16.

9. The SOP expression to be implemented using a decoder design is: $y = \Sigma(0, 7, 8, 9, 10)$. Which one of the following ICs must be used to implement the expression?
   - A. 7442.
   - B. 74138.
   - C. 74139.
   - D. 74154.

10. What type of external gating must be used with the above decoder to implement the expression given in question 9?
    - A. A 16-bit NAND.
    - B. A 4-bit NAND.
    - C. A 5-bit NAND.
    - D. A 5-bit NOR.

11. How many 1-of-16 decoder chips must be used to implement ten logic functions of the same four input variables.
    - A. One.
    - B. Four.
    - C. Ten.
    - D. Sixteen.

12. The address decoder in Figure 3-37 can be used to decode decimal address 10,155 by NORing/ORing the following output lines:
    - A. 10, 1, 5, and 5.
    - B. 1, 0, 1, 5, and 5.
    - C. 2, 7, 10, and 11.
    - D. This circuit cannot be used to decode this address.

13. To use the 7442 as a 1-of-10 demultiplexer, you must apply the input logic, D_{in} to:
    - A. The enable input line.
    - B. The most significant input line.
    - C. The least significant input line.
    - D. The 7442 cannot be used as a 1-of-10 demultiplexer.

14. What type of circuit is integrated into a ROM to reduce the number of required address lines?
    - A. Multiplexer.
    - B. PLA.
    - C. Encoder.
    - D. Decoder.

15. When a ROM is disabled, its data output lines:
    - A. Generate a constant logic 0 state.
    - B. Generate a constant logic 1 state.
    - C. Are open with infinite resistance.
    - D. ROM devices cannot be enabled/disabled.

16. You want to use a ROM to implement a logic design that has four outputs that use the same eight input variables. To implement this design, you must use a:
    A. 256 × 4 ROM.
    B. 16 × 8 ROM.
    C. 4 × 8 ROM.
    D. 8 × 4 ROM.

17. A ROM device has 12 address pins and 8 data pins. This ROM is a:
    A. 2K × 8 ROM.
    B. 256 × 12 ROM.
    C. 32K bit ROM.
    D. None of these.

18. One-shot ROMs, or PROMs, are programmed by:
    A. Storing a charge on a floating gate.
    B. Blowing a fuse link.
    C. Making connections with a metallization mask.
    D. Electrically altering previously stored information.

19. The internal gate structure of a PLA is:
    A. AND/OR.
    B. NAND/NOR.
    C. XOR/XNOR.
    D. AND only.

20. The SOP expression generated by the y_3 output in Figure 3-58 on Page 3-85 is:
    A. $y_3 = \bar{A}B\bar{C} + A\bar{B}\bar{C}$.
    B. $y_3 = \overline{ABC} + \bar{B}C + ABC$.
    C. $y_3 = ABC + ABC + ABC$.
    D. None of these.

EXAMINATION ANSWERS

1. C — A multiplexer is a data selector.

2. C — A 100-to-1 multiplexer circuit must have seven channel select lines.

3. B — Two 74153 4-to-1 multiplexer ICs are required to make an 8-to-1 multiplexer.

4. C — Using the obvious solution for combinational logic design with multiplexers, a circuit with four input variables requires a 16-to-1 multiplexer.

5. B — Using the n-1 solution for four input variables requires an 8-to-1 multiplexer.

6. B — Four input pairs would be generated for three input variables using the n-1 multiplexer design solution.

7. A — Another name for a 3-line-to-8-line decoder is a 1-of-8 demultiplexer.

8. C — A 1-of-8 decoder is required to implement a design of three input variables.

9. D — A 74154.

10. C — A 5-bit NAND must be used with a decoder to implement the expression given in question 9.

11. A — One 1-of-16 decoder chip must be used to implement ten logic functions of the same four input variables.

12. C — The address decoder in Figure 3-37 can be used to decode decimal address 10,155 by NORing/ORing the following output lines: 2, 7, 10, and 11.

13. B — To use the 7442 as a 1-of-10 demultiplexer, you must apply the input logic, D_{in}, to the most significant input line.

14. D — A decoder circuit is integrated into a ROM to reduce the number of required address lines.

15. C — When a ROM is disabled, its data output lines are open with infinite resistance.

16. A — You want to use a ROM to implement a logic design that has four outputs that use the same eight input variables. To implement this design, you must use a 256 × 4 ROM.

17. C — A ROM device has 12 address pins and 8 data pins. This ROM is a 32K bit ROM. Note: $2^{12} \times 8 = 32{,}768 = 32K$.

18. B — One-shot ROMs, or PROMs, are programmed by blowing a fuse link.

19. A — The internal gate structure of a PLA is AND/OR.

20. A — The SOP expression generated by the y_3 output in Figure 3-58 is $y_3 = \overline{A}B\overline{C} + A\overline{B}\,\overline{C}$.

UNIT 4

SEQUENTIAL LOGIC DESIGN

CONTENTS

Introduction . . . . Page 4-3
Unit Objectives . . . . Page 4-4
Unit Activity Guide . . . . Page 4-5
Flip-Flops . . . . Page 4-6
Sequential Circuits: Registers and Counters . . . . Page 4-32
Designing Sequential Circuits . . . . Page 4-70
Summary . . . . Page 4-92
Unit Examination . . . . Page 4-93
Examination Answers . . . . Page 4-98

INTRODUCTION

Your learning journey through this course is providing you with the fundamental building blocks of a digital computer. You will find that all digital computers use both combinational and sequential logic circuits to execute a program and make decisions. In the last three units, you learned about the decision making devices used in digital electronics. Digital circuits employing decision making elements are called combinational logic circuits. Now, you are about to explore the other half of the digital world and learn about circuits that incorporate memory elements into their operation. Recall that such circuits are called **sequential circuits**.

Since memory is the basic element of any sequential circuit, this unit will begin with a discussion of semiconductor memory elements called **flip-flops**. Once you learn about flip-flops, you will be able to understand the operation of the two fundamental types of sequential circuits called **registers** and **counters**. You will learn how these circuits operate and how to design special sequential circuits to satisfy any given application. Now, let's get to work.

UNIT OBJECTIVES

When you complete this unit, you should be able to:

1. Construct an RS flip-flop using both NAND and NOR gates.
2. Explain the operation of an RS, D, and JK flip-flop.
3. Explain how flip-flops are used to remember, or store, binary data.
4. Sketch timing diagrams for flip-flop circuits.
5. Design frequency divider circuits.
6. Design clock generating circuits.
7. Explain the operation of buffer registers, shift registers, and counters.
8. Design serial-to-parallel and parallel-to-serial converter circuits.
9. Describe the features of several common IC buffer registers, shift registers, and counters.
10. List the major steps required to design sequential circuits.
11. Develop state tables for sequential circuits.
12. Design special purpose sequential circuits using both D and JK flip- flops.
13. Explain when output logic must be designed for a sequential circuit.

UNIT ACTIVITY GUIDE

☐ Read section on "Flip-Flops."

☐ Answer Self-Test Review Questions 1-13.

☐ Read section on "Sequential Circuits: Registers and Counters."

☐ Answer Self-Test Review Questions 14-28.

☐ Perform Experiments 11 and 12 in Unit 8.

☐ Read section on "Designing Sequential Circuits."

☐ Answer Self-Test Review Questions 29-34.

☐ Study Summary.

☐ Perform Experiment 13 in Unit 8.

☐ Complete Unit Examination.

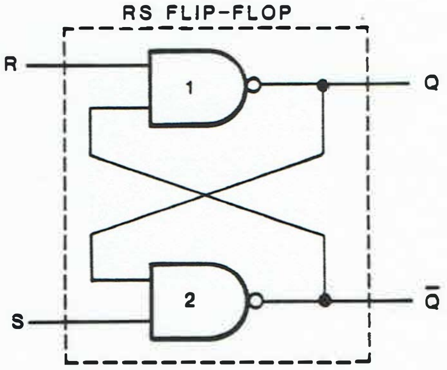
☐ Check Examination Answers.

FLIP-FLOPS

In the last three units, you discovered that the basic decision making element of a combinational logic circuit is the logic gate. In the same way, the **flip-flop** is the basic **memory** element used in a sequential logic circuit. A flip-flop is a digital device that stores, or remembers, a given logic state, 1 or 0. It will remain in a given state until it is made to change by the application of one or more external signals. There are three types of flip-flops used in semiconductor memory and sequential logic circuits. They are the **RS, D, and JK** flip-flops.

The RS Flip-Flop

You have already had some experience with an RS flip-flop in Unit 1. Do you remember the cross-coupled NAND gate circuit shown in Figure 4-1? There are two inputs labeled R and S, and two outputs labeled Q and $\overline{Q}$. The Q output is fed-back to NAND gate 2, while the $\overline{Q}$ output is fed-back to NAND gate 1. The corresponding truth table shows how the circuit operates. Let's see if we can verify its operation.



R	S	Q	$\overline{Q}$	STATE
0	0	1	1	RACE
0	1	1	0	SET
1	0	0	1	RESET
1	1	LAST		HOLD

Figure 4-1

An RS flip-flop is constructed using cross-coupled NAND gates.

Suppose a logic 0 is applied to the R input and a logic 1 is applied to the S input. Recall that a NAND gate will always generate a logic 1 output when any of its inputs are low. Consequently, since R is low, the output of gate 1 must be high, regardless of its other input value. As a result, Q = 1. Now, this logic 1 state is fed-back and applied to gate 2. In addition, we have assumed that S is high. Since both inputs to NAND gate 2 are high, it will generate a logic 0 output. In summary, an input of RS = 01 produces outputs of Q = 1 and $\overline{Q}$ = 0. This is called the **SET state**, since the Q output is a logic 1.

Next, suppose that an input value of RS = 10 is applied to the circuit. This time, the output of gate 2 must be high, since one of its inputs, namely S, is low. The high output of gate 2 is fed-back and applied to gate 1. The other input to gate 1 is R, which is high in this case. Since both inputs to gate 1 are high, it generates a low output. To summarize, an input value of RS = 10 produces outputs of Q = 0 and $\overline{Q}$ = 1. This is called the **RESET state**, since the Q output generates a logic 0.

The most important state of the circuit is the **HOLD**, or **memory state**. In this state, the Q and $\overline{Q}$ outputs remain unchanged from their previous states. In other words, the circuit remembers the state it was in last. For this reason, the hold state is also referred to as the **last state** condition. Notice from the truth table that this state occurs when the input value is RS = 11.

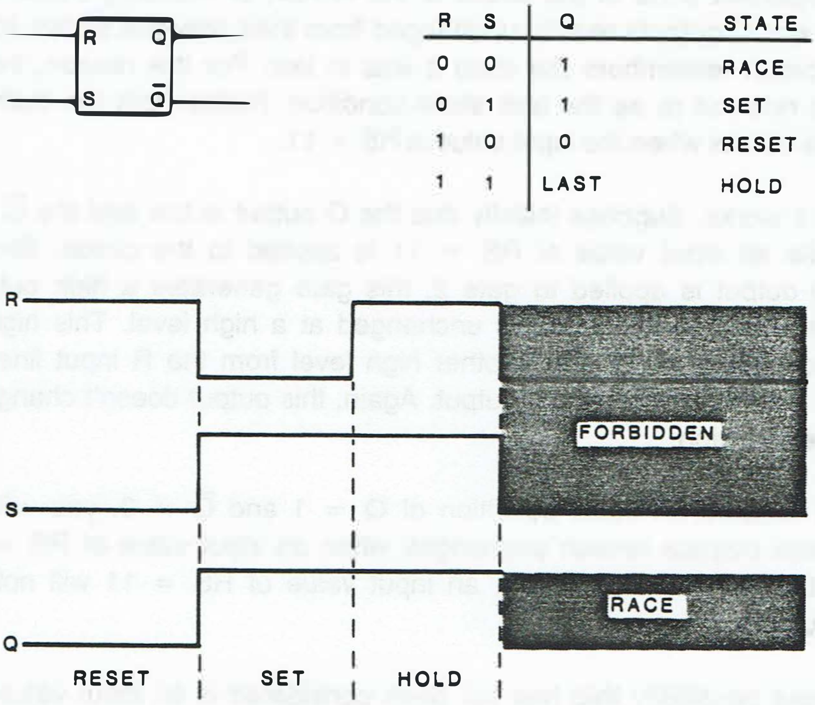
Here's how it works. Suppose initially that the Q output is low and the $\overline{Q}$ output is high, while an input value of RS = 11 is applied to the circuit. Since the low level Q output is applied to gate 2, this gate generates a high output. In other words, the $\overline{Q}$ output remains unchanged at a high level. This high level is applied to gate 1 along with another high level from the R input line. As a result, gate 1 generates a logic 0 output. Again, this output doesn't change from the last state condition.

If you next assume an initial condition of Q = 1 and $\overline{Q}$ = 0, you will again find that these outputs remain unchanged when an input value of RS = 11 is applied to the circuit. In summary, an input value of RS = 11 will not affect the circuit outputs.

The only input possibility that has not been considered is an input value of RS = 00. Notice from the truth table that this is called the **RACE state**. If both inputs to the circuit are low, both outputs try to go high, since any low input to a NAND gate produces a high output. This is where the term **race condition** comes from. However, you don't know the result of the race until the inputs change. If only one input goes high, the flip-flop enters a SET or RESET state. That is easy to determine. On the other hand, if both inputs go high, the race concludes with unpredictable results — one output goes high and the other goes low. You have no way to determine the outcome of the race, since this is a function of the switching times of the integrated transistors and the input impedance of the gates. For that reason, the race condition is sometimes called the **forbidden** or **invalid** state.

There is one other reason why the RACE state should be avoided. Flip-flops, by their nature, are considered bistable devices. That is, when the Q output is high, the $\overline{Q}$ output should be low, and vice versa. Thus, by definition, the RACE state is illegal since both outputs are high in that state. The solution to the problem is found in the D flip-flop to be discussed shortly.

The black-box symbol in Figure 4-2 is used to represent an RS flip-flop. The associated truth table and timing diagram summarize the operation of this device. Notice that the output state is always with reference to the Q output. Of course, you know that the $\overline{Q}$ output is just the opposite, or complement, of the Q output, except in the RACE state.

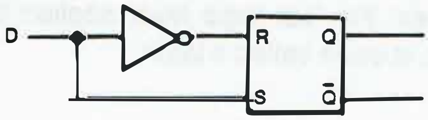


R	S	Q	STATE
0	0	1	RACE
0	1	1	SET
1	0	0	RESET
1	1	LAST	HOLD

Figure 4-2
The symbol and operation of an RS flip-flop.

The D Flip-Flop

The inherent racing problem of an RS flip-flop can be solved by connecting an inverter to the R input line as shown in Figure 4-3. The single input line is labeled **D**. Here, the race condition can never occur since the R and S inputs are never in the same logic state. By knowing the operation of an RS flip-flop, you should be able to verify the associated truth table. Notice that if D is low, the Q output resets. Conversely, the Q output sets when D is high.



D	R	S	Q	OUTPUT
0	1	0	0	RESET
1	0	1	1	SET

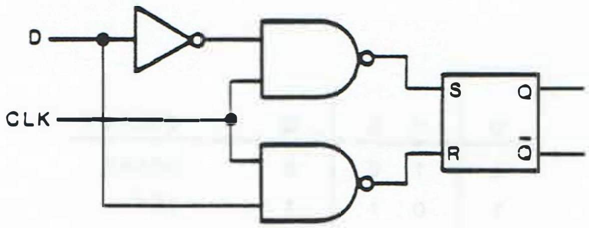
Figure 4-3

The racing problem is solved by connecting an inverter to one of the input lines.

Although the circuit in Figure 4-3 solves the racing problem, it also eliminates the most important output state of a flip-flop: the hold, or memory, state. To keep the hold state, you must add two NAND gates to the circuit as shown in Figure 4-4. The D input line drives one side of each NAND and a common line labeled **CLOCK**, or **CLK**, drives the other side of each gate. Let's see how this affects the circuit operation.

According to the associated truth table, if CLK is low, or inactive, it doesn't matter what D is, and the circuit output remains unchanged from its last state. To see this, if CLK is low, the outputs of both NAND gates must be high. From your knowledge of an RS flip-flop, you know that this results in the HOLD state. Now, if CLK is high, or active, the circuit output "follows" the logic applied to the D input. Consequently, with CLK high, a logic 0 input on D results in the RESET state and a logic 1 input on D results in the SET state. The circuit in Figure 4-4 is called a **D flip-flop**, for obvious reasons.

The operation of a D flip-flop is controlled by the clock input. When the clock is active, the Q output follows the D input. When the clock goes inactive, the Q output holds, or latches, the last logic level applied to the flip-flop input. For this reason, the D flip-flop is often called a latch.



CLK	D	R	S	Q	STATE
0	X	1	1	LAST	HOLD
1	0	1	0	0	RESET
1	1	0	1	1	SET

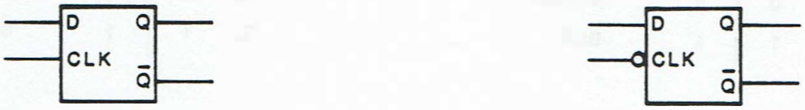
Figure 4-4

The addition of two NAND gates restores the memory state and creates a D flip-flop.

Two symbols for a D flip-flop are shown in Figure 4-5. Notice that the CLK line is bubbled in the one symbol. As you can see from the associated truth tables and timing diagrams, a bubbled CLK line results in an active low condition for this line. In other words, the flip-flop outputs can only change when a low level is applied to a bubbled CLK line. The operation of the D flip-flops in Figure 4-5 can be summarized by the following two statements:

1. When the CLK line is active, the logic applied to the D input is transferred to the Q output.
2. When the CLK line is inactive, the outputs do not change, regardless of the logic applied to D.

Apply these statements to the timing diagrams in Figure 4-5. Are they correct?



CLK	D	Q	STATE
0	X	LAST	HOLD
1	0	0	RESET
1	1	1	SET

CLK	D	Q	STATE
1	X	LAST	HOLD
0	0	0	RESET
0	1	1	SET

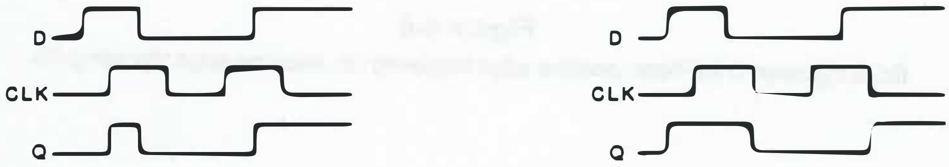
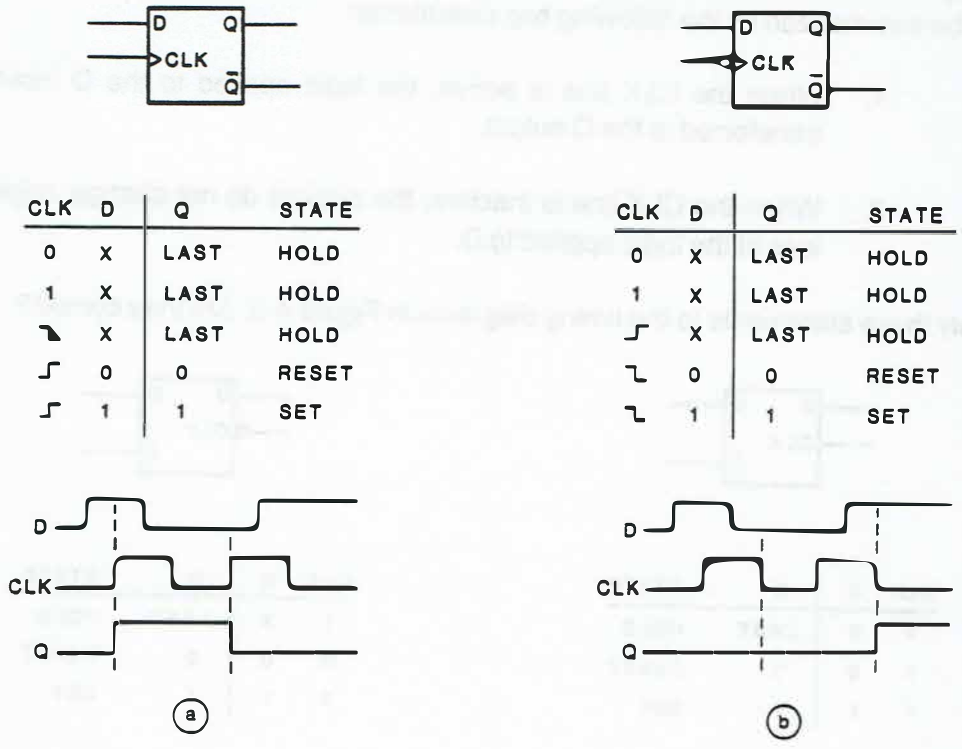


Figure 4-5
Two symbols for a level triggered D flip-flop.

LEVEL VS. EDGE TRIGGERING

The D flip-flops shown in Figure 4-5 are called **level triggered**, since the CLK line is active for a given logic level. The disadvantage of this is that the flip-flop output can change any time during the active level of the clock line. As long as the CLK line is active, the output follows the input.

An alternative to level triggering is **edge triggering**. With edge triggering, the flip-flop output can only change during the rising or falling edge of the CLK signal. A flip-flop can be **positive edge** triggered or **negative edge** triggered. Both are shown in Figure 4-6.



CLK	D	Q	STATE
0	X	LAST	HOLD
1	X	LAST	HOLD
↓	X	LAST	HOLD
↑	0	0	RESET
↑	1	1	SET

CLK	D	Q	STATE
0	X	LAST	HOLD
1	X	LAST	HOLD
↑	X	LAST	HOLD
↓	0	0	RESET
↓	1	1	SET

Figure 4-6

Edge triggered D flip-flops: positive edge triggering (a), negative edge triggering (b).

The symbol in Figure 4-6a is positive edge triggered, since the output can only change on the rising, or low-to-high, edge of the CLK signal. In Figure 4-6b, the output can only change on the falling, or high-to-low, edge since this flip-flop is negative edge triggered.

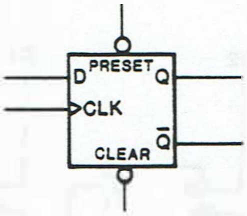
Edge triggering is provided by integrating a small resistor/capacitor, or RC, differentiator circuit in series with the CLK input line. When a flip-flop uses edge triggering, a small notch is placed on the symbol next to the CLK input line as shown in the figure. In addition, a bubble is added when the flip-flop operates using negative edge triggering.

From the truth tables you can see that small clock edge symbols are used to designate the rising and falling edges of the CLK signal. Notice that a constant logic level on the CLK line results in the hold state. The outputs can only change when an active **transition** occurs on the CLK line. You should study and compare the truth tables and timing diagrams in Figures 4-5 and 4-6 until you feel comfortable with the different clocking options available with D flip-flops.

PRESET AND CLEAR

The last feature that might be required in a D flip-flop circuit is a **PRESET** and/or **CLEAR** function. When power is first applied to a circuit, the flip-flop outputs are unpredictable and random. Many times your design application will require that all the flip-flops be initially cleared to a logic 0 or preset to a logic 1 state when power is applied to the circuit. To accomplish this, some flip-flops include two lines labeled PRESET and CLEAR.

A positive edge triggered D flip-flop with PRESET and CLEAR is shown in Figure 4-7, along with its associated truth table. Here, the PRESET and CLEAR functions are active low, as indicated by the bubbled lines. From the truth table, you see that a logic 0 applied to the PRESET line causes the output to go high, or set, regardless of what is taking place on the D and CLK input lines. Likewise, a logic 0 applied to the CLEAR line causes the Q output to go low, or clear.



PRESET	CLEAR	CLK	D	Q	STATE
0	1	X	X	1	PRESET
1	0	X	X	0	CLEAR
1	1	↓	X	LAST	HOLD
1	1	↑	0	0	RESET
1	1	↑	1	1	SET
0	0	X	X	?	RACE

Figure 4-7
A positive edge triggered D flip-flop with PRESET and CLEAR.

The PRESET and CLEAR functions are top priority, since the flip-flop responds to these operations regardless of what is taking place on the D and CLK input lines. In other words, PRESET and CLEAR override all the other flip-flop inputs. This is why the D and CLK inputs are labeled as don't cares (X's) in the truth table when PRESET and CLEAR are active.

In practice, the PRESET and CLEAR lines are activated temporarily to set or clear the flip-flop to an initial state. Then they are deactivated to allow for normal operation of the device. However, note from the truth table that PRESET and CLEAR cannot be activated at the same time, since this results in a race condition.

Example 4-1

Four D flip-flops are shown connected by a common clock line in Figure 4-8. The four outputs of the flip-flops generate a 4-bit word ***Q*** *= Q3 Q2 Q1 Q0. The circuit input is the 4-bit word* ***D*** *= D3 D2 D1 D0. This circuit is called a 4-bit* ***buffer register****, or* ***latch****. Determine the latch output for the following conditions:*

a. *Power on and CLK high.*
b. ***D*** *= 0100 and CLK high.*
c. ***D*** *= 0100 and CLK falling.*
d. ***D*** *= 1111 and CLK rising.*
e. ***D*** *= 0000 and CLK falling.*

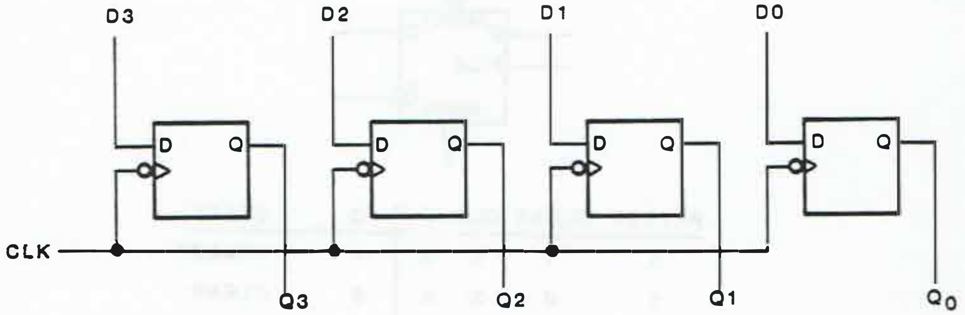


Figure 4-8

A 4-bit buffer register (Example 4-1).

a. When the power is applied, the flip-flop outputs are unpredictable and in a random state.

b. The outputs will remain in a random state since the CLK is being held inactive at a constant high level.

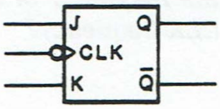
c. The output will equal the input, or **Q** = **D** = 0100, since CLK is active during its falling edge.

d. The output will remain unchanged, or **Q** =0100, since CLK is inactive during its rising edge.

e. The output will follow the input, or **Q** =**D** = 0000, since CLK is active again.

The JK Flip-Flop

The last flip-flop that you need to learn about is the JK flip-flop. As you will soon discover, this device is particularly important for designing counting circuits. A typical JK flip-flop symbol is shown in Figure 4-9. First, notice that it has two inputs, J and K, and two outputs, Q and $\overline{Q}$. In addition to the J and K inputs, there is a CLK input. From the symbol, you can tell that the CLK line is negative edge triggered.



CLK	J	K	Q	STATE
↑	X	X	LAST	HOLD
X	0	0	LAST	HOLD
↓	0	1	0	RESET
↓	1	0	1	SET
↓	1	1	$\overline{\text{LAST}}$	TOGGLE

Figure 4-9

A negative edge triggered JK flip-flop.

From the associated truth table, you find that the JK generates all of the output states available from a D flip-flop and, in addition, includes an output state called the **TOGGLE** state. If the JK input is 00, the outputs of the flip-flop remain unchanged from their previous state. Like the D flip-flop, this state is called the HOLD state. If CLK is active and the JK inputs are 01, the Q output resets. If CLK is active and JK = 10, the Q output sets.

Up to this point, the operation of the JK is similar to a D flip-flop. However, unlike a D flip-flop, the JK output can be made to **toggle**. This happens when the JK inputs are both high, or 11. In digital jargon, toggling means to "change state". Thus, if a given output was in a logic 0 state and the flip-flop is toggled, the output would change to a logic 1 state. Conversely, a logic 1 output would toggle to a logic 0.

Two conditions must be present for the flip-flop outputs to toggle: the JK inputs must be 11, and an active CLK transition must occur. Study and learn (memorize if necessary) the JK truth table in Figure 4-9! You will find it extremely valuable when analyzing counter circuits in the next section.

Example 4-2

Suppose a continuous CLK signal is applied to the flip-flop in Figure 4-10 as shown.

a. *Sketch the Q output signal.*

b. *How does the frequency of this output signal compare to the CLK frequency?*

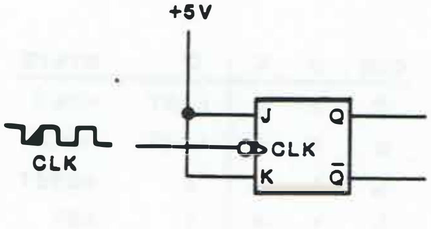


Figure 4-10
A JK flip-flop hardwired for the toggling state (Example 4-2).

c. *The duty cycle of a digital signal is the ratio of the logic 1 time during a cycle to the signal period, times 100 percent, or:*

Duty Cycle = [(Logic 1 Time)/Period] × 100%

Suppose the duty cycle of the CLK signal were 25%. How does this affect the duty cycle of the flip-flop output signal?

a. Since the JK inputs are tied high, the flip-flop is in its toggling mode and the outputs will toggle each time an active transition of the CLK signal occurs. From the JK symbol, you see that the flip-flop is negative edge triggered. Thus, the output will change state for each successive falling edge of the CLK signal. The resulting timing diagram is shown in Figure 4-11.

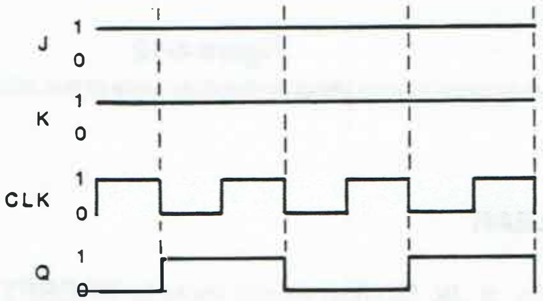


Figure 4-11

A timing diagram for Figure 4-10 (Example 4-2).

b. From the timing diagram, you can see that for every two CLK cycles there is only one output cycle. Thus, the frequency of the output is one-half the frequency of the CLK signal. Because of this frequency relationship, a JK flip-flop in its toggling mode is sometimes called a divide-by-2, or ÷2, circuit.

c. The duty cycle of the CLK signal has no effect on the duty cycle of the toggling flip-flop output. The output signal will always be symmetrical, resulting in a 50% duty cycle. The timing diagram in Figure 4-12 illustrates this idea. Here, the CLK signal has a 25% duty cycle, but the duty cycle of the JK output remains 50%. You might want to sketch timing diagrams using other CLK duty cycles to convince yourself of this relationship.

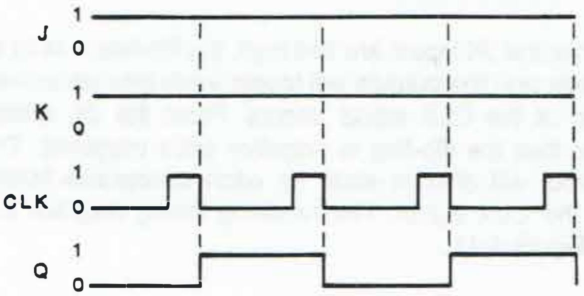
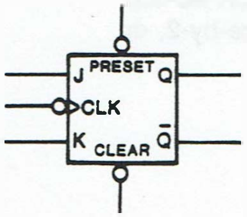


Figure 4-12

The CLK duty cycle has no effect on the duty cycle of the JK output (Example 4-2).

PRESET AND CLEAR

Like the D flip-flop, a JK flip-flop might include PRESET and CLEAR features as shown in Figure 4-13. When this is the case, these lines are usually active low, as indicated by the bubbles. Remember, that PRESET and CLEAR always take top priority. When either one of these lines is activated, the J, K, and CLK inputs have no effect on the device. Of course, PRESET and CLEAR must not be activated at the same time.



PRESET	CLEAR	CLK	J	K	Q	STATE
0	1	X	X	X	1	PRESET
1	0	X	X	X	0	CLEAR
1	1	↑	X	X	LAST	HOLD
1	1	↓	0	0	LAST	HOLD
1	1	↓	0	1	0	RESET
1	1	↓	1	0	1	SET
1	1	↓	1	1	$\overline{\text{LAST}}$	TOGGLE
0	0	X	X	X	?	RACE

Figure 4-13

A negative edge triggered flip-flop with PRESET and CLEAR.

FLIP-FLOP TIMING SPECIFICATIONS

There are three specifications that you must be aware of when designing with flip-flop circuits. They are: **propagation delay time, setup time,** and **hold time**. These timing specifications are particularly important when flip-flops are used in high speed circuits.

Propagation Delay Time (t_p)

Recall that propagation delay for a logic gate was a measure of the time it takes the output of a gate to change due to a change on the gate inputs. In other words, propagation delay is a measure of a logic gate's speed. Likewise, the propagation delay of a flip-flop is a measure of its speed. The propagation delay of a flip-flop is a measure of the time it takes the output to change after an active state is detected on the CLK line. As an example, if an edge triggered JK flip-flop data sheet indicates that t_p = 12 nS, it takes twelve nanoseconds for its output to change after the active edge of the CLK pulse.

Setup Time (t_s)

This specification is unique to clocked devices such as flip-flops. The setup time of a flip-flop is the amount of time that logic must be present on its input lines before an active CLK state arrives. If the setup time of a D flip-flop is specified at t_s = 20 nS, this means that logic applied to the D input line must be present at least twenty nanoseconds before it can be clocked to the output. With a JK device, the setup time specifies the minimum time that data must be present on **both** the J and K inputs before an active clock pulse arrives.

Hold Time (t_h)

Hold time is similar to setup time except that it deals with how long logic must be held on the flip-flop input lines after clocking. A JK flip-flop specifying a hold time of 10 nS requires that any logic applied to its J and K inputs must be held a minimum of ten nanoseconds after the CLK line is activated.

Some Real Devices

Now that you are familiar with the operation of flip-flops, let's take a look at some popular flip-flop ICs.

THE 7474 D FLIP-FLOP

The 7474, shown in Figure 4-14, is a dual edge triggered D flip-flop with PRESET and CLEAR. Notice from the symbol that the IC contains two independent positive edge triggered flip-flops. Furthermore, the PRESET and CLEAR lines are active low. When these lines are not in use, they should be tied high.

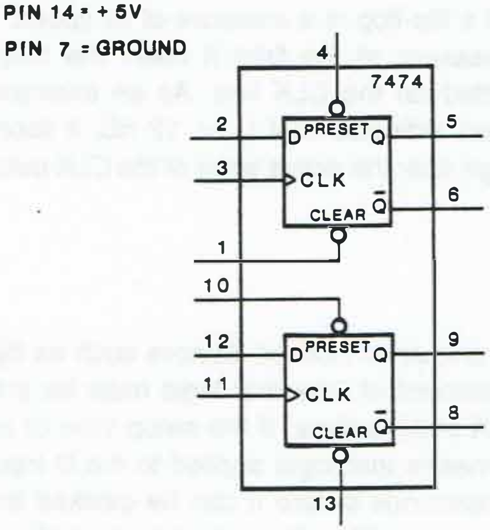


Figure 4-14

The 7474 dual D flip-flop.

THE 7475 D FLIP-FLOP

As you can see from Figure 4-15, the 7475 contains four D flip-flops. For this reason, the 7475 is sometimes called a quad, or 4-bit, latch. The flip-flops are positive level triggered and clocked in pairs. That is, the flip-flops are clocked by a logic 1 level, and a given CLK line is used to clock two flip-flops simultaneously. To use the device as a 4-bit latch, you simply connect the two CLK lines together. Observe also that the this device does not include any preset or clear features.

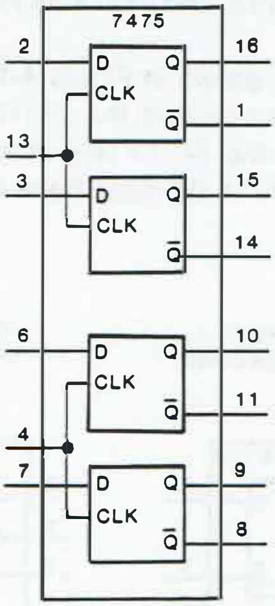


Figure 4-15
A 7475 quad latch.

THE 7473 and 7476 JK FLIP-FLOPS

These two devices are shown in Figure 4-16. Each device contains two independent JK flip-flops. They are both clocked by a low level. The major difference between the two is that the 7476 includes both the preset and clear features, while the 7473 only includes the clear feature.

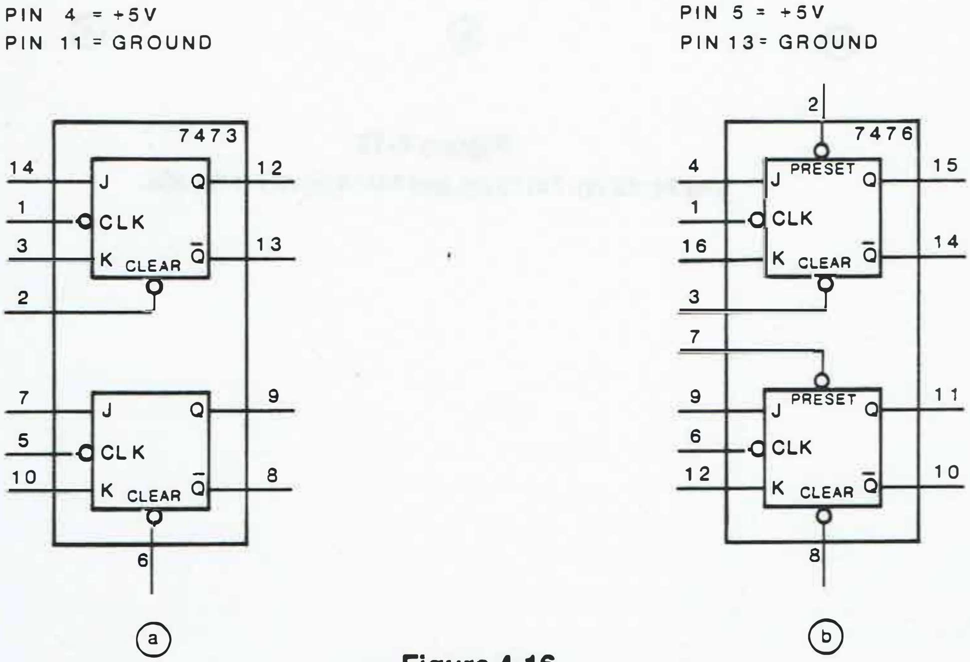


Figure 4-16
The 7473 (a) and 7476 (b) JK flip-flop ICs.

THE 74112, 74113, and 74114 JK FLIP-FLOPS

These devices, shown in Figure 4-17, each contain two negative edge triggered JK flip-flops. Notice that the 74112 and 74114 both include preset and clear features, while the 74113 only includes a preset feature. Furthermore, observe that the two 74114 flip-flops have a common CLK line and a common CLEAR line.

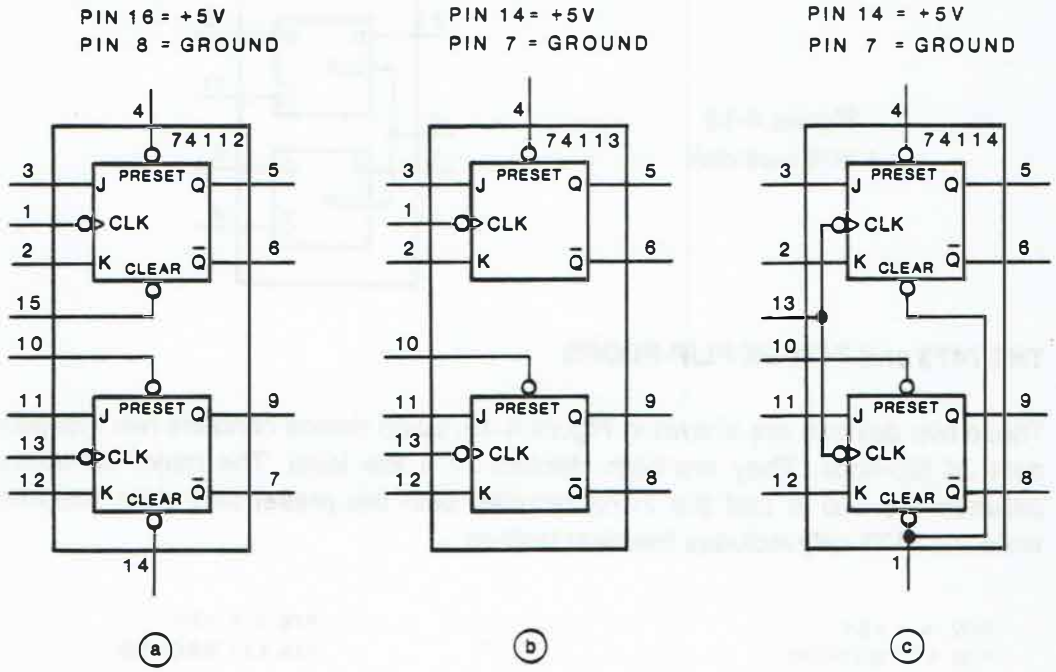


Figure 4-17
The 74112 (a), 74113 (b), and 74114 (c) JK flip-flop ICs.

A Clock Generator IC, The 555 Timer

Before you leave flip-flops, it is a good idea to take a look at a device that can be used to generate clock signals. Several such devices are available, but the 555 timer is by far the most popular.

The 555 is really a linear (analog) IC that is used to generate digital pulses. A typical 555 circuit is shown in Figure 4-18. Here, the IC is connected to a 5-volt supply since we are going to use it to generate a TTL clock signal. Actually, this device can be operated using a supply anywhere from +4 to +15 volts.

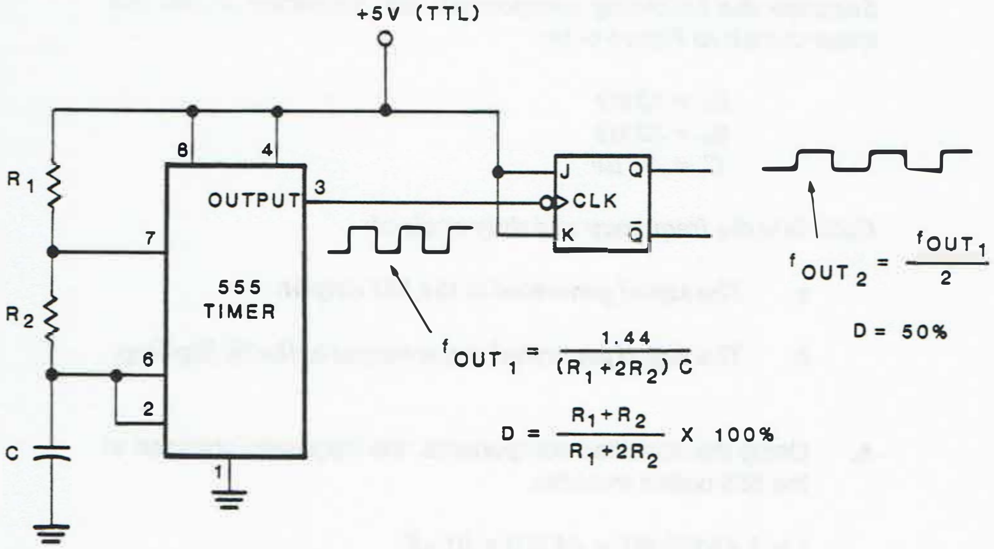

+5V (TTL)
R1
R2
C
8
4
7
6
2
1
3
OUTPUT
555 TIMER
J
Q
CLK
K
Q̄
$f_{OUT_1} = \frac{1.44}{(R_1+2R_2)C}$
$D = \frac{R_1+R_2}{R_1+2R_2} \times 100\%$
$f_{OUT_2} = \frac{f_{OUT_1}}{2}$
D = 50%

Figure 4-18

A 555 clock generator circuit.

Aside from the supply connections, notice that the 555 requires a minimum of three external components to operate: two resistors (R_1 and R_2) and a capacitor, C. When connected as shown in Figure 4-18, the 555 will generate a continuous rectangular output signal with the following characteristics:

Frequency: $f = 1.44/(R_1 + 2R_2) \times C$

Duty Cycle: $D = [(R_1 + R_2)/(R_1 + 2R_2)] \times 100\%$

The resistor values can range from 1 Kilohm to 3.3 Megohms, while the value of the capacitor (C) can be anything above 1000 pF. However, the formulas assume the resistor values are in kilohms and the capacitor values are in microfarads. Various combinations of these components allow the 555 to generate frequencies anywhere between 1 Hz and 1 MHz.

Most digital circuits require a clock signal with a 50% duty cycle. However, for a given frequency, it is often difficult to obtain a 50% duty cycle from a 555 IC using standard value resistor and capacitor components. To achieve a 50% duty cycle, a JK flip-flop can be added to the 555 output as shown in the figure. Here, the 555 output clocks a JK in its toggling mode. Recall that a JK flip-flop connected in its toggling mode will always generate a 50% duty cycle, regardless of the CLK duty cycle. However, the frequency obtained from the flip-flop will be one-half the 555 output frequency.

Example 4-3

Suppose the following components are connected to the 555 timer circuit in Figure 4-18:

$R_1 = 10\,k\Omega$
$R_2 = 22\,k\Omega$
$C = .01\,\mu F$

Calculate the frequency and duty cycle of:

a. The signal generated at the 555 output.

b. The signal generated at the output of the JK flip-flop.

a. Using the specified components, the frequency obtained at the 555 output must be:

$f = 1.44/(10\text{ k}\Omega + 44\text{ k}\Omega) \times .01\,\mu\text{F}$

$= 2.66\text{ kHz}$

The duty cycle is:

$D = [(10\text{ k}\Omega + 22\text{ k}\Omega)/(10\text{ k}\Omega + 44\text{ k}\Omega)] \times 100\%$

$= 59.2\%$

b. At the flip-flop output, the 555 output frequency is divided by two, or 2.66 kHz/2 = 1.33 kHz. The duty cycle obtained from a JK flip-flop in its toggling mode is always 50%.

I should mention that there are several more ways to use the 555 timer other than that shown in Figure 4-18. This versatile little device can be used to generate single, one shot, pulses as well as continuous signals. There are many books that describe the operation and application of the 555 and other timer/generators.

Self-Test Review

1. How can you determine the outputs of a flip-flop when it is in a race state?

____

____

2. Develop a truth table for an RS flip-flop that has been constructed using cross coupled NOR gates.

3. Another name for a D flip-flop is a ______________.

4. How can you make a D flip-flop from a JK flip-flop? (Hint: Look at how a D flip-flop was constructed from an RS flip-flop in Figure 4-3.)

5. Design an 8-bit latch using two 7475 D flip-flops.

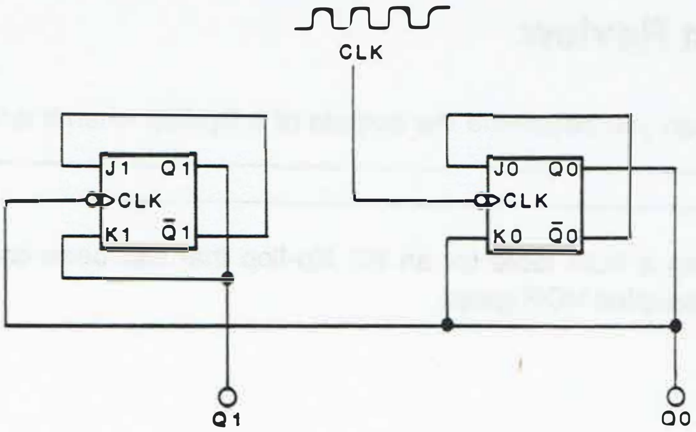


Figure 4-19

A JK flip-flop circuit (Question 6).

6. Assume that both flip-flops in Figure 4-19 are initially reset. Sketch the timing diagrams for the Q0 and Q1 outputs for four active clock pulses.

7. Design a divide-by-16 circuit using JK flip-flops.

8. Assume the input clock frequency to the circuit in question 7 is 960 Hz. What is the frequency generated at each of the four flip-flop outputs?

   A. 1st flip-flop: ____________
   B. 2nd flip-flop: ____________
   C. 3rd flip-flop: ____________
   D. 4th flip-flop: ____________

9. If the flip-flops used in question 7 have a propagation delay time of 25 nS, how long will it take the last flip-flop to toggle after an active clock signal is applied to the first flip-flop? ____________

10. Let **Q** = Q3 Q2 Q1 Q0 in Figure 4-20. Assuming the flip-flops are initially cleared, determine the first four **Q** words due to four successive clock pulses.

   A. 1st clock: Q = ____________
   B. 2nd clock: Q = ____________
   C. 3rd clock: Q = ____________
   D. 4th clock: Q = ____________

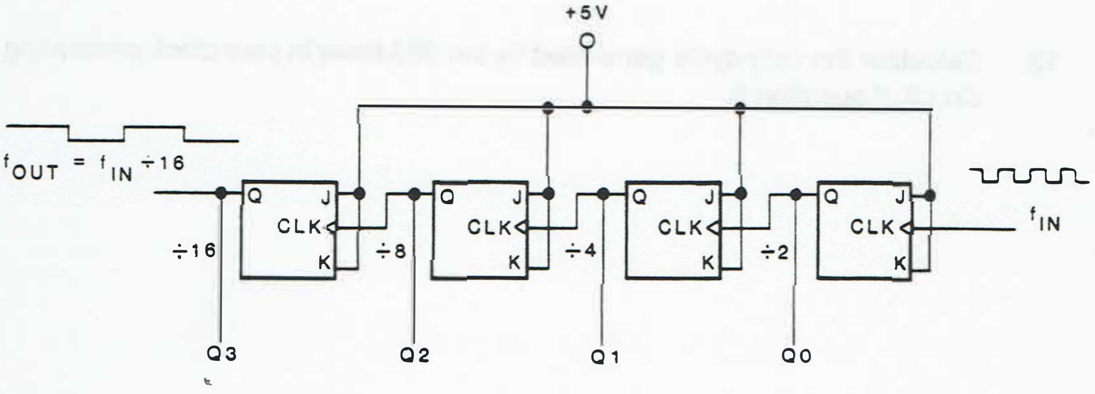


Figure 4-20
A ÷ 16 circuit (Question 10).

11. Looking at the flip-flop outputs as a succession of Q words as you obtained in question 10, what is the circuit doing? ______________________

____

12. Design a 555 timer circuit to generate a 10 KHz clock signal with a 50% duty cycle.

13. Calculate the duty cycle generated by the 555 timer in your clock generating circuit of question 9.

Answers

1. You can't determine the outputs of a flip-flop when it is in a race condition because they are unpredictable.

2.

R S	Q	Output State
0 0	Last	HOLD
0 1	1	SET
1 0	0	RESET
1 1	?	RACE

3. Latch.

4. See Figure 4-21 on Page 4-30.

5. See Figure 4-22 on Page 4-30.

6. See Figure 4-23 on Page 4-30.

7. See Figure 4-20 on Page 4-27.

8A. 1st flip-flop: 480 Hz
8B. 2nd flip-flop: 240 Hz
8C. 3rd flip-flop: 120 Hz
8D. 4th flip-flop: 60 Hz

9. 4 × 25 ns, or 100 ns

10A. 1st clock: **Q** = 0001
10B. 2nd clock: **Q** = 0010
10C. 3rd clock: **Q** = 0011
10D. 4th clock: **Q** = 0100

11. The circuit is counting up in binary.

12. Refer to Figure 4-24 and the solution that follows on Page 4-31.

13. 86.11%.

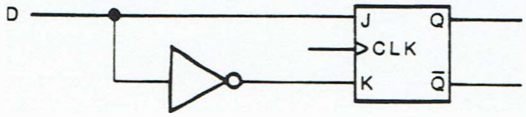


Figure 4-21

A JK flip-flop modified to function as a D flip-flop (answer to question 4).

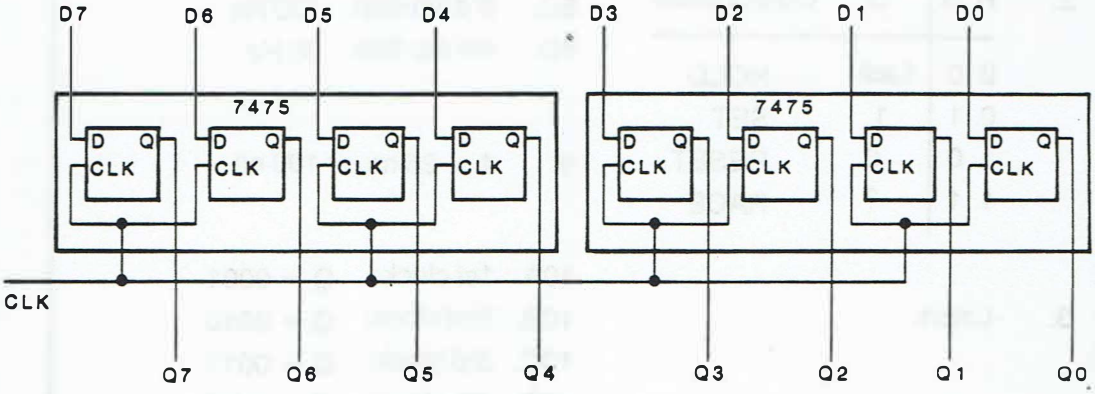


Figure 4-22

An 8-bit latch using two 7475's (answer to question 5).

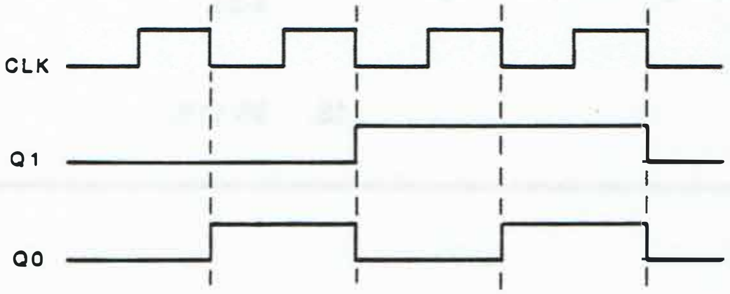


Figure 4-23

Timing diagram for Figure 4-21 (answer to question 6).

Solution to Questions 12 and 13.

12. To design this circuit, pick convenient values for R_2 and C, then calculate R_1. With a little algebra, you can solve the 555 frequency equation for R_1 as follows:

$$R_1 = [1.44/(f_{OUT} \times C)] - 2R_2$$

Now, let R_2 = 1000 ohms, and C = 0.01 μF. The value of f_{OUT} must be twice the desired frequency, since the JK flip-flop divides the 555 output frequency by two. Thus, you must let f_{OUT} = 20 kHz. Using these values in the equation, you get R_1 = 5200 ohms.

These resistor and capacitor values generate a duty cycle of approximately 86%. To generate a 50% duty cycle, you must add a JK flip-flop to the timer output. The flip-flop divides the timer frequency by two to generate the desired 10 KHz frequency with a 50% duty cycle.

13. The duty cycle in Figure 4-24 is:

$$D = [(R_1 + R_2)/(R_1 + 2R_2)] \times 100\%$$

$$= 86.11\%$$

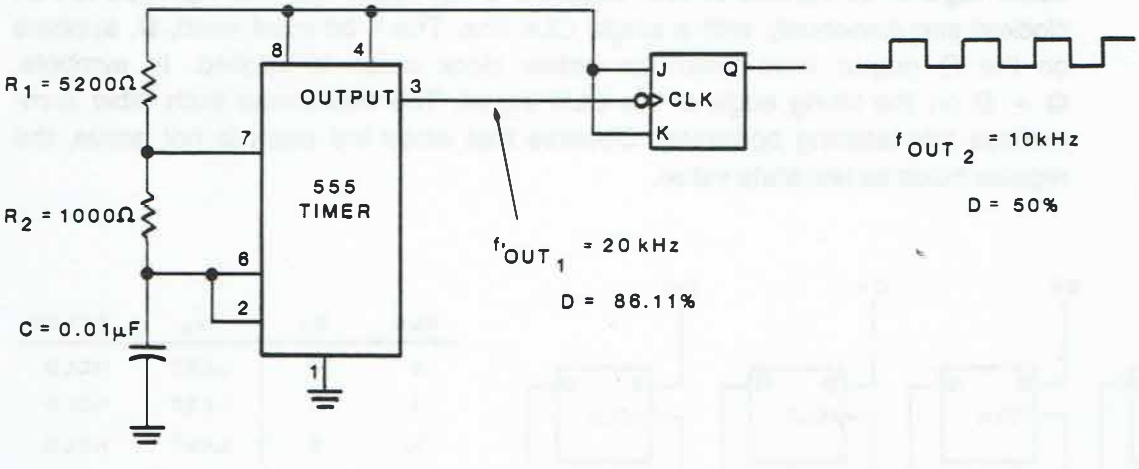


Figure 4-24

A 555 clock generating circuit to generate a frequency of 10 kHz with a 50% duty cycle (answer to question 12).

SEQUENTIAL CIRCUITS: REGISTERS AND COUNTERS

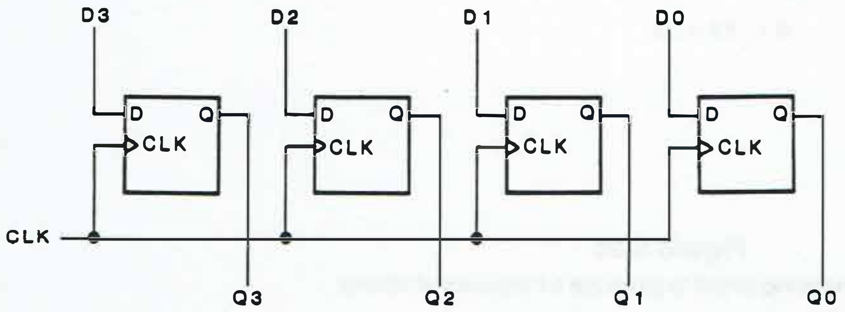
The most common types of sequential circuits are registers and counters. Registers and counters are constructed using the flip-flops that you learned about in the last section. Thus, it is easy to understand the operation of these circuits if you know how the various flip-flops operate.

A **register** is simply a collection of single-bit flip-flop memory elements. Most registers are constructed using RS or D flip-flops. The flip-flops are connected together to store a multi-bit binary word. A register which simply stores, or latches, a binary word is called a **buffer register**, or **latch**. On the other hand, a register which loads and modifies a multi-bit word by shifting the bits right or left is called a **shift register**.

A counter is a sequential circuit that "counts" clock pulses. Each time an active clock pulse arrives, the counter increments or decrements a value stored in its memory elements. The new count value is then held until another clock pulse arrives. In most cases, counters are constructed using JK flip-flops. Now, lets see what makes these circuits tick.

Buffer Registers or Latches

The simplest form of a buffer register is shown in Figure 4-25. It is a 4- bit buffer register comprised of four separate D flip-flops. The four flip-flops are all clocked simultaneously with a single CLK line. The 4-bit input word, **D**, appears on the **Q** output lines when an active clock pulse is applied. In symbols, **Q** = **D** on the rising edge of the CLK signal. The associated truth table summarizes this latching operation. Observe that when the clock is not active, the register holds its last state value.

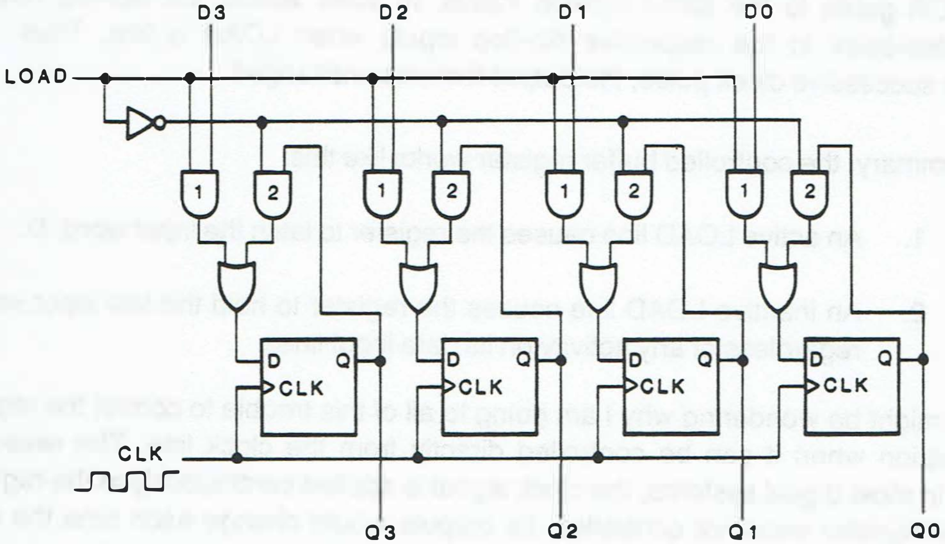


CLK	D_n	Q_n	STATE
0	X	LAST	HOLD
1	X	LAST	HOLD
↓	X	LAST	HOLD
↑	0	0	LATCH
↑	1	1	LATCH

Figure 4-25

A 4-bit buffer register.

A second kind of buffer register is the **controlled buffer register**. This register allows you to control precisely when an input value is to be latched. A 4-bit controlled buffer register is shown in Figure 4-26. Again, four D flip-flops are connected via a common clock line. In addition to the flip-flops, an AND/OR gate combination is used to control the application of the input value, **D**, to the flip-flops.



LOAD	D_n	Q_n	STATE
0	X	LAST	HOLD
1	0	0	LATCH
1	1	1	LATCH

Figure 4-26

A 4-bit controlled buffer register.

Here's how a controlled buffer register works. A continuous clock signal is applied to the CLK line. If LOAD is active (high in Figure 4-26), the **D** input word bits are applied through their respective AND gates to the flip-flop inputs. In Figure 4-26, a high level on the LOAD line causes a high level to be applied to one side of each of the gates labeled 1. Recall that if one input line of a 2-bit AND gate is high, the gate output reflects the logic at the other input. Thus, the input bits (D0 through D3) are passed through their respective AND gates to one side of each OR gate. The outputs of the AND gates labeled 2 are logic 0, since they see a logic 0 input via an inverter on the LOAD line. As a result, each OR gate will pass the **D** input logic from the AND gates labeled 1 to each respective flip-flop input. The next clock pulse will cause the **D** input logic to be latched, or loaded, into the register.

Next, suppose LOAD is inactive, or low, in Figure 4-26. Here, a low level on the LOAD line causes a logic 0 to be applied to the AND gates labeled 1 and a logic 1 to be applied to the AND gates labeled 2. This time, the **D** input logic is blocked from the circuit, since the AND gates labeled 1 are disabled. However, the outputs from the AND gates labeled 2 will reflect the logic of their other input line. Notice that the **Q** output lines of the flip-flops are applied to the other side of each of the AND gates labeled 2. As a result, the AND gates labeled 2 will reflect the **Q** logic of the flip-flop outputs. This logic is then passed through the OR gates to the same flip-flop inputs. In other words, the flip-flop outputs are **fed-back** to the respective flip-flop inputs when LOAD is low. Thus, with each successive clock pulse, the output remains unchanged.

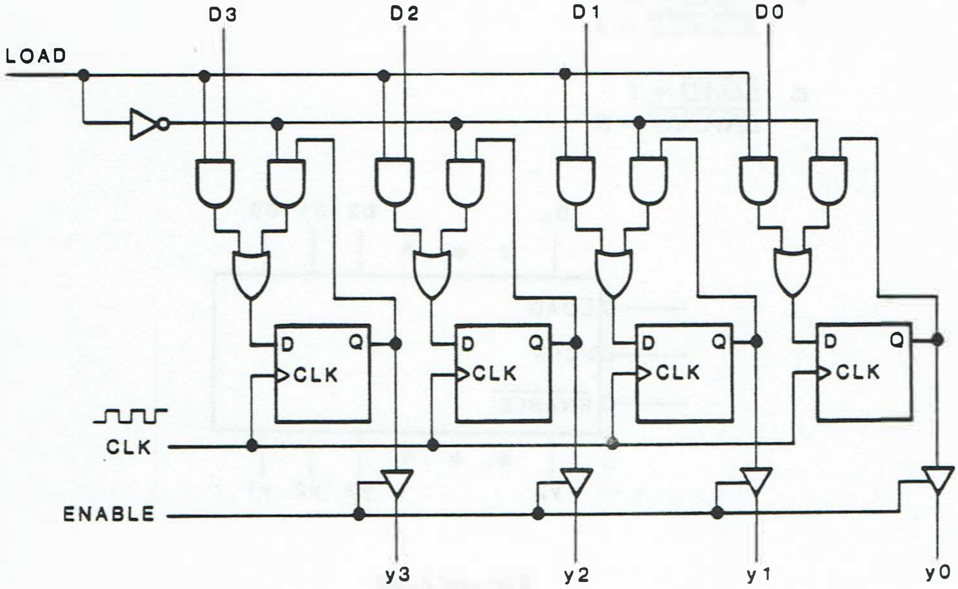
In summary, the controlled buffer register works like this:

1. An active LOAD line causes the register to latch the input word, **D**.
2. An inactive LOAD line causes the register to hold the last input value, regardless of any activity on its data input lines.

You might be wondering why I am going to all of this trouble to control the register operation when it can be controlled directly from the clock line. The reason is that in most digital systems, the clock signal is applied continuously to the register. If the register were not controlled, its outputs would change each time the input bits changed and an active clock pulse arrived. This is undesirable, especially when the register input bits are being supplied from a set of data bus lines that are common to other registers in the system. The idea with a controlled buffer register is to activate the register via its LOAD line only when it is necessary to latch, or load, a given data value.

TRI-STATE OUTPUTS

The register in Figure 4-27 shows how one additional feature can be added to a controlled buffer register. Here, the register output lines are three-state buffered. Consequently, the **Q** output logic only appears on the **y** output lines when ENABLE is active. When ENABLE is inactive, an open circuit (infinite resistance) is seen at the register outputs. Again, this feature is especially desirable in bus systems.



ENABLE	LOAD	D_n	y_n	STATE
0	X	X	∞	OPEN
1	0	X	LAST	HOLD
1	1	0	0	LATCH
1	1	1	1	LATCH

Figure 4-27

A controlled buffer register with 3-state outputs.

Example 4-4

The symbol for a controlled buffer register with three-state outputs is shown in Figure 4-28. Suppose that **D** *= 0101 and a continuous clock signal is being applied to the register. What will the register outputs be under the following set of sequential circumstances:*

a. $\overline{LOAD} = 1$
$\overline{ENABLE} = 1$

b. $\overline{LOAD} = 0$
$\overline{ENABLE} = 1$

c. $\overline{LOAD} = 1$
$\overline{ENABLE} = 0$

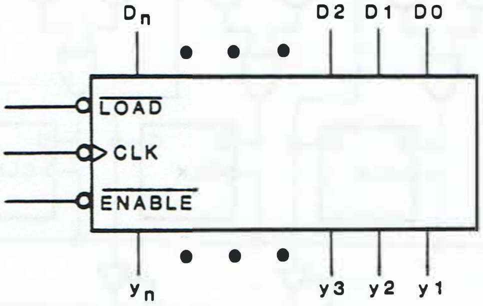


Figure 4-28

A typical controlled buffer register with 3-state outputs (Example 4-4).

a. First, notice that both control lines, $\overline{\text{LOAD}}$ and $\overline{\text{ENABLE}}$, are active low as indicated by the bar over the line name. Since both are high at this time, the register outputs will be in their infinite resistance, or open, state.

b. Next, $\overline{\text{LOAD}}$ is active and $\overline{\text{ENABLE}}$ is inactive. As a result, the register outputs are still in their open state. However, the **D** input value is latched, or loaded, in the register as a result of the active $\overline{\text{LOAD}}$ line.

c. Here, $\overline{\text{LOAD}}$ is inactive and $\overline{\text{ENABLE}}$ is active. Thus, the contents of the register will appear on the output lines. Since the value 0101 was loaded in part b, the register output is **y** = 0101.

SOME REAL DEVICES

I have picked the 74LS374 and 74LS377 as representative of IC buffer registers. Both are 8-bit registers in a single IC package. The 74LS374, shown in Figure 4-29a, has three-state outputs but is not a controlled buffer register. On the other hand, the 74LS377 shown in Figure 4-29b is controlled but does not have three-state outputs. Notice in both cases, that the respective control lines are active low. This is typical of most IC control lines. Other IC buffer registers are listed in Table 4-1 for your information.

Table 4-1: IC Buffer Registers

Device Number	Function
74LS173	4-bit Controlled Buffer Tri-state
74LS174	6-bit Buffer with Clear
74LS175	4-bit Buffer with Inverted Outputs Available
74LS273	8-bit Buffer with Clear
74LS374	8-bit Buffer with Tri-state
74LS377	8-bit Controlled Buffer
74LS378	6-bit Buffer
74LS379	4-bit Buffer with Inverted Outputs Available

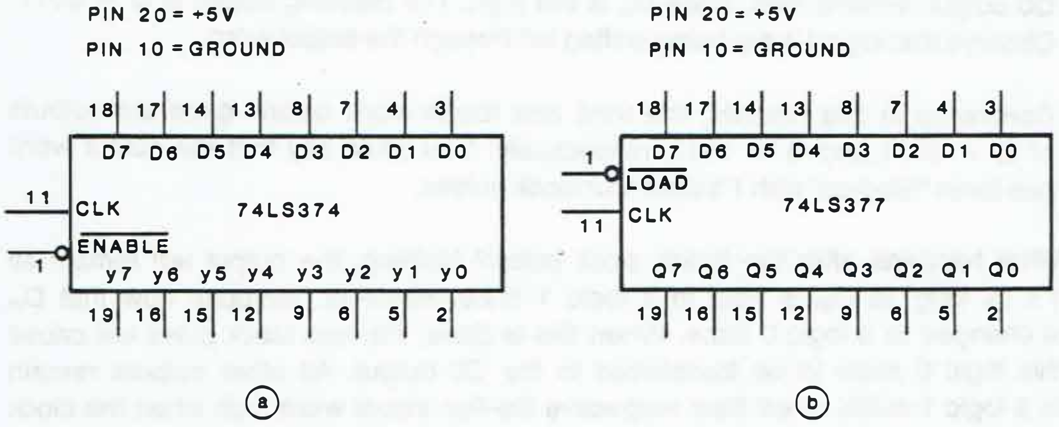


Figure 4-29
Two typical buffer registers: the 74LS374 (a), and 74LS377 (b).

Shift Registers

A shift register is a special kind of buffer register that shifts the bits stored in the register left or right. Shift registers are used in computers to perform arithmetic operations on binary numbers, and to convert between serial and parallel data formats. More about this later.

Aside from shifting bits either left or right, shift registers can be classified as either **serial input** or **parallel input**. We'll describe the serial input shift register first.

SERIAL INPUT SHIFT REGISTERS

Serial input means one bit at a time. A 4-bit serial input shift register is shown in Figure 4-30. Observe that a serial input line, labeled D_{in}, drives the least significant flip-flop. Then, the output of each flip-flop drives the input of each succeeding flip-flop. To see how it works, suppose $D_{in} = 1$ and the register is initially cleared, or **Q** = 0000. When the first clock pulse hits, the logic 1 applied to D_{in} line is transferred to the Q0 output. The Q1 through Q3 outputs remain low since their respective flip-flop inputs were low when the clock arrived. As a result, the output is **Q** = 0001 after the first clock pulse.

When the second clock pulse hits, the Q1 output goes to a logic 1, since its flip-flop input was high as a result of the previous Q0 output. In addition, the Q0 output remains high, since D_{in} is still high. The resulting output is **Q** = 0011. Observe that logic 1's are being shifted left through the output word.

Continuing in this manner, the third and fourth clock pulses generate outputs of **Q** = 0111 and **Q** = 1111, respectively. You could say that the output word has been "filled-up" with 1's after four clock pulses.

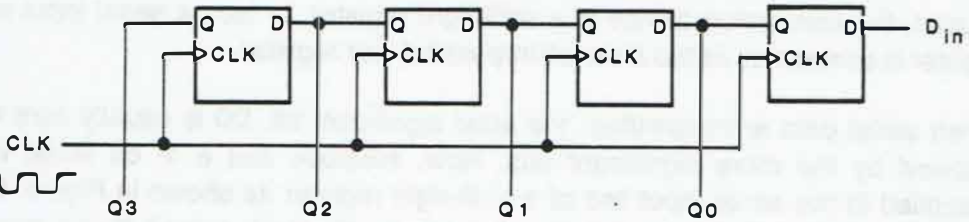
What happens after the fourth clock pulse? Nothing, the output will remain all 1's as long as D_{in} is held in a logic 1 state. However, suppose now that D_{in} is changed to a logic 0 state. When this is done, the next clock pulse will cause this logic 0 state to be transferred to the Q0 output. All other outputs remain in a logic 1 state, since their respective flip-flop inputs were high when the clock pulse arrived. The resulting output is **Q** = 1110. If D_{in} is held in a logic 0 state, the next three clock pulses will produce outputs of **Q** = 1100, 1000, and 0000, respectively. Here, 0's are being shifted left, or to say it another way, the output word is filling-up with logic 0's.

Example 4-5

Assume that the serial input line to a 4-bit shift register is held high and the shift register is initially cleared.

a. *What would the register outputs be for four successive clock pulses if the serial input were shifted **right**?*

b. *How could the shift-left register in Figure 4-30 be made into a shift-right register?*



D_{in}	Q3	Q2	Q1	Q0
0	Q2	Q1	Q0	0
1	Q2	Q1	Q0	1

Figure 4-30

A 4-bit, serial input, shift-left register.

a. Assuming the shift register is initially cleared and the serial input is to be shifted right, four successive clock pulses will produce the following outputs:

1st clock:	Q = 1000
2nd clock:	Q = 1100
3rd clock:	Q = 1110
4th clock:	Q = 1111

b. The simplest way to make the shift-left register into a shift-right register is to reverse the output lines. That is, Q0 is switched with Q3 and Q1 is switched with Q2. When this is done, Q3 becomes the least significant output line, followed by Q2, followed by Q1, followed by Q0 as the most significant output line.

SERIAL-TO-PARALLEL CONVERSION

In digital systems, data is often transferred in serial, one bit at a time. This is done especially over long distances to eliminate unnecessary data lines. Only two transmission lines are required for serial data transmission: a single data line and a ground line. On the other hand, parallel transmission requires a separate data line for each data bit in a multi-bit word, plus a ground line.

Recall that all computers internally process data in parallel form. For that reason, serial data received by a computer must be converted to parallel before it can be processed. A shift-right operation is normally used to accomplish this conversion. The shifting operation can be done using software or hardware. If hardware is used, the conversion device is a shift-right register. In fact, a serial input shift register is sometimes called a **serial-in/parallel-out** register.

When serial data is transmitted, the least significant bit, D0 is usually sent first, followed by the more significant bits. Now, suppose that a 4- bit serial word is applied to the serial input line of a shift-right register as shown in Figure 4-31. What would the register output be after the fourth clock pulse? If you thought 0101, you were right. Notice how the shift-right register has converted the 4-bit serial word into a 4-bit parallel word after four clock pulses. Of course, the application of the serial input bits must be synchronized with the clock pulses to assure that the setup and hold time requirements of the flip-flops are satisfied.

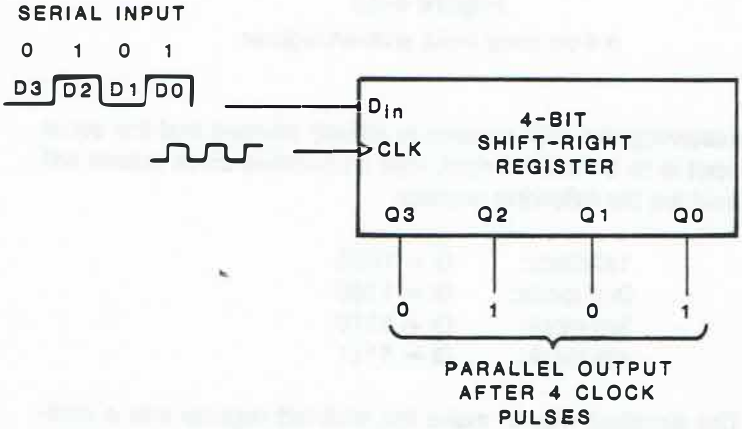

SERIAL INPUT
0 1 0 1
D3 D2 D1 D0
D_{in}
CLK
4-BIT SHIFT-RIGHT REGISTER
Q3 Q2 Q1 Q0
0 1 0 1
PARALLEL OUTPUT AFTER 4 CLOCK PULSES

Figure 4-31
A hardware serial-to-parallel converter using a shift-right register.

PARALLEL INPUT SHIFT REGISTERS

A parallel input shift register is shown in Figure 4-32. Look familiar? It should, since it is very similar to the controlled buffer register discussed earlier. Notice the control line labeled LOAD/$\overline{\text{SHIFT}}$. It regulates the parallel data load and data shift operations of the register. When this line is high, the AND gates labeled 1 are enabled and pass the parallel input word, **D**, to the flip-flop inputs. When a clock pulse hits, the input value is loaded into the register, or **Q** = **D**.

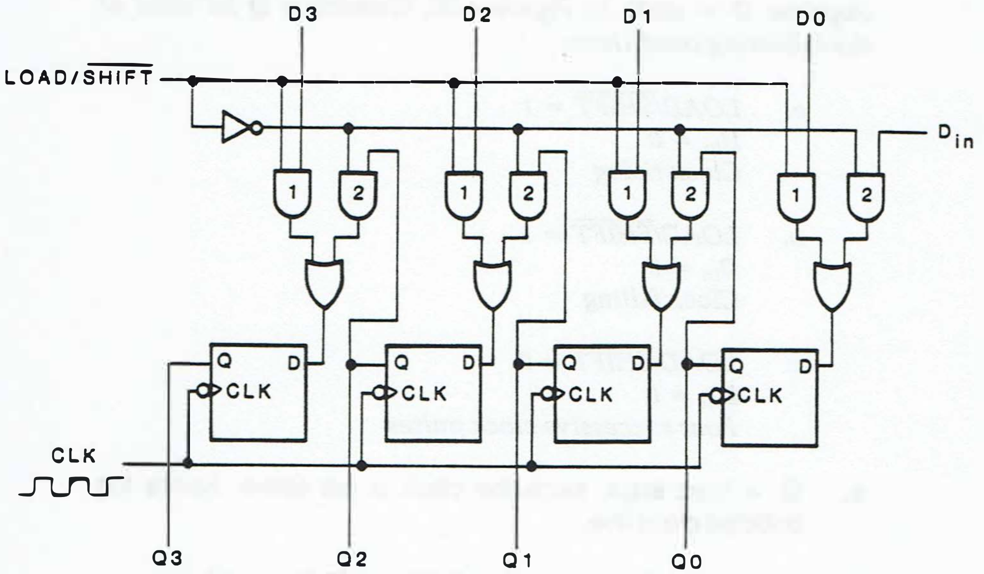


Figure 4-32

A 4-bit parallel-input shift register.

Now, suppose the LOAD/$\overline{\text{SHIFT}}$ line goes low. This allows the output logic of each flip-flop to pass through the AND gates labeled 2 to the input of the next succeeding flip-flop. Thus, as long as the LOAD/$\overline{\text{SHIFT}}$ line is held low, the value loaded in the register is shifted left one bit position for each clock pulse. Of course, the logic state existing on the D_{in} line is shifted into the least significant bit position.

The register operation can, therefore, be summarized as follows:

1. When LOAD/$\overline{\text{SHIFT}}$ is high, the value of **D** is loaded, or latched, into the register and **Q** = **D** when a clock pulse arrives.

2. When LOAD/$\overline{\text{SHIFT}}$ is low, the value in the register is shifted-left one bit position for each clock pulse. The logic on the D_{in} line is always shifted into the least significant bit position.

You also know how to make the register in Figure 4-32 operate as a shift- right register. Simply reverse the output lines.

Example 4-6

*Suppose **D** = 0101 in Figure 4-32. Determine Q for each of the following conditions:*

a. *LOAD/$\overline{SHIFT}$ = 1*
$D_{in} = 0$
Clock rising

b. *LOAD/$\overline{SHIFT}$ = 1*
$D_{in} = 0$
Clock falling

c. *LOAD/$\overline{SHIFT}$ = 0*
$D_{in} = 0$
Four successive clock pulses.

a. **Q** = Last state, since the clock is not active. Notice the bubbled clock line.

b. **Q** = **D** = 0101, since both LOAD and CLK are active.

c. Since SHIFT is active, the value loaded in part b will be shifted left one bit position for each clock pulse. Since, $D_{in} = 0$, a logic 0 is shifted into the least significant bit position. The four clock pulses generate the following outputs:

1st clock: **Q** = 1010
2nd clock: **Q** = 0100
3rd clock: **Q** = 1000
4th clock: **Q** = 0000

It is obvious why the register in Figure 4-32 could be called a parallel-in/parallel-out register. However, this register can also be made to operate like a serial-in/parallel-out register. How? By simply storing an initial value of 0000 into the register, then applying the serial input logic to the D_{in} line. For this reason, it is sometimes called a serial/parallel input register.

PARALLEL-TO-SERIAL CONVERSION

When a digital computer system transmits serial information, it must first convert its internal parallel data to serial data. The least significant data bit, D0, is transmitted first, followed by the more significant data bits. This can be done using software or hardware techniques. When hardware is used, the conversion device is a parallel input shift-right register.

The 4-bit shift-right register shown in Figure 4-33 illustrates this application. The word to be transmitted is applied to the parallel input lines of the register. The word is then loaded by activating the LOAD line. The least significant output line, Q0, is used as the serial output line. As soon as the word is loaded, the least significant bit appears on this line. Thus, the LOAD operation actually transmits the least significant data bit. The remaining bits are transmitted by activating the register $\overline{\text{SHIFT}}$ line. Each time the register shifts, the next bit will appear on the Q0 output line. The $\overline{\text{SHIFT}}$ line is held in an active state until all the data bits have been transmitted.

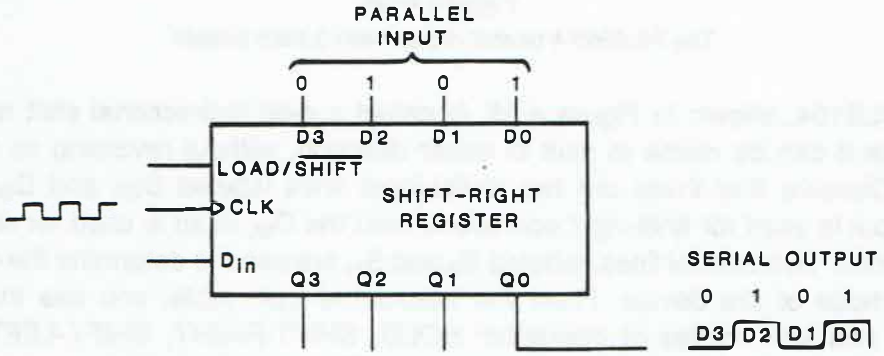


Figure 4-33

A hardware parallel-to-serial converter using a shift-right register.

The register in Figure 4-33 could also be used as a serial-to-parallel converter by utilizing the D_{in} line as you observed earlier. Now, let's take a look at two typical shift register ICs.

SOME REAL DEVICES

The 74LS395, shown in Figure 4-34, is a serial/parallel input 4-bit shift register. In addition to the operation discussed earlier, you will find that this device includes a clear feature and three-state outputs. The device is used as shown for shift-left operations. To act as a shift-right register, the outputs must be reversed as discussed earlier.

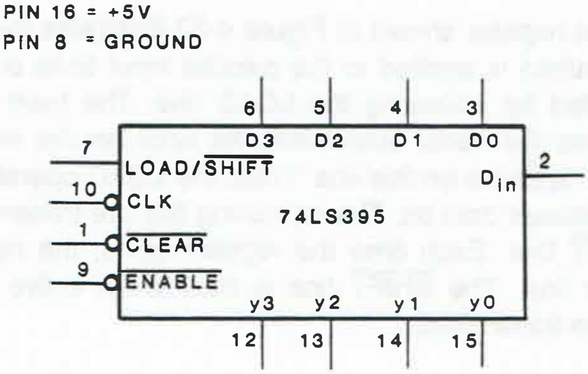
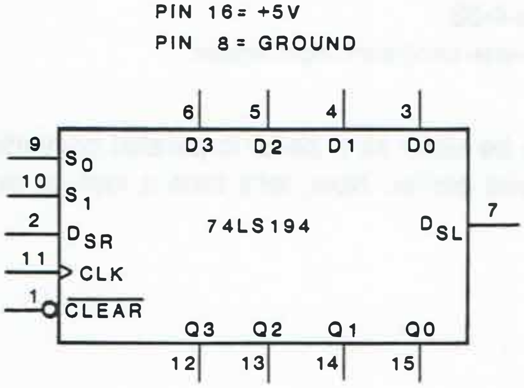


Figure 4-34
The 74LS395 4-bit shift register with 3-state outputs.

The 74LS194, shown in Figure 4-35, is called a 4-bit bidirectional shift register because it can be made to shift in either direction, without reversing its output lines. Observe that there are two serial-input lines labeled D_{SR} and D_{SL}. The D_{SR} input is used for shift-right operations, and the D_{SL} input is used for shift-left operations. Two control lines, labeled S_0 and S_1, are used to determine the operational mode of the device. From the associated truth table, you see that this device has four modes of operation: HOLD, SHIFT-RIGHT, SHIFT-LEFT, and LOAD. Each of these operations should be familiar to you by now.



S_1	S_0	OPERATION
0	0	HOLD
0	1	SHIFT-RIGHT
1	0	SHIFT-LEFT
1	1	LOAD

Figure 4-35
The 74LS194 4-bit bidirectional shift register.

There are many more different shift register ICs available, some of which are listed in Table 4-2. A brief look at this Table should give you a "feel" for the commercial availability of shift register devices.

Table 4-2: Several Popular Shift Register ICs

Device Number	Function
74LS95	4-bit Parallel Input
74LS164	8-bit Serial Input with Clear
74LS194	4-bit Bidirectional Parallel Input with Clear
74LS195	4-bit Parallel Input with Clear
74LS295	4-bit Parallel Input with Tri-State Outputs
74LS299	8-bit Bidirectional Parallel Input with Tri-State Outputs and Clear
74LS395	4-bit Parallel Input with Tri-State Outputs and Clear

Counters

One of the most useful sequential circuits in digital electronics is the counter. As stated earlier, a counter simply counts the clock pulses applied to the clock input of the circuit. Thus, the input to the counter is a series of clock pulses and the counter output is a multi-bit binary count value. Most counters count in straight binary, but there are some that use special output codes.

Counters also perform an additional function of frequency division. Recall that a JK flip-flop divides its clock frequency by two. Since most counters are constructed using JKs, various clock frequency divisions are available from the counter outputs. This is why a counter is sometimes referred to as a divide-by-n counter, where n is the frequency divisor.

In this text, you will study three fundamental counter circuits: the **ripple counter, synchronous counter,** and **ring counter.** These basic counting circuits are used in most of the commercially available counter ICs. The ripple and synchronous counter use JK flip-flops, while the ring counter uses D flip-flops.

THE RIPPLE COUNTER

A 3-bit ripple counter is shown in Figure 4-36. Before trying to analyze any counting circuit, always look at how the individual circuit flip-flops are being clocked. Notice in Figure 4-36 that the least significant flip-flop, 0, is being clocked by the external clock signal. The output of this flip-flop is then used to clock flip-flop 1. The output of flip-flop 1 clocks flip-flop 2. In addition, all the flip-flops are negative edge triggered.

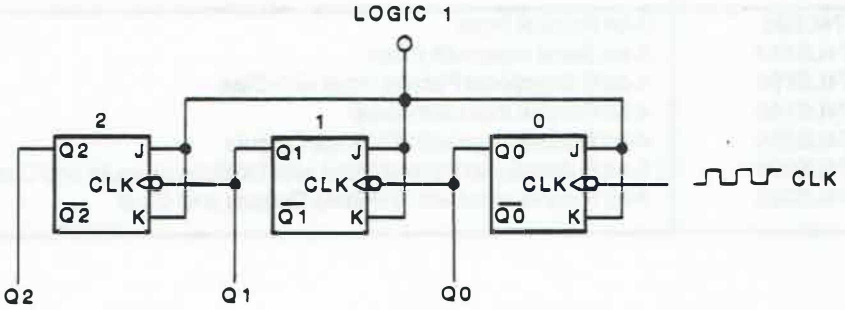


Figure 4-36

A 3-bit ripple counter.

The next thing you must do is determine the state that each flip-flop is in before it is clocked. This determination is relatively easy for the ripple counter in Figure 4-36, since each flip-flop is hardwired in its toggling state. As a result, a given flip-flop will toggle each time it is clocked.

Let's analyze the ripple counter one bit at a time. Assume that the counter output is initially cleared, or **Q** = 000. Now, flip-flop 0 will toggle each time an external clock pulse hits. Therefore, its output will be as follows for eight consecutive clock pulses:

	Q2	Q1	Q0
Initial state:	0	0	0
1st clock:			1
2nd clock:			0
3rd clock:			1
4th clock:			0
5th clock:			1
6th clock:			0
7th clock:			1
8th clock:			0

Next, let's determine the Q1 output. Since flip-flop 1 is being clocked by Q0, the Q1 output will toggle each time the Q0 output makes a negative, or 1 to 0, transition. Filling in the Q1 output column of our table, using this idea, you get:

	Q2	Q1	Q0
Initial state:	0	0	0
1st clock:		0	1
2nd clock:		1	0
3rd clock:		1	1
4th clock:		0	0
5th clock:		0	1
6th clock:		1	0
7th clock:		1	1
8th clock:		0	0

Notice that Q1 toggles each time Q0 makes a high-to-low transition. Next, what do you suppose Q2 will do? Your right! The Q2 output will toggle each time Q1 makes a high-to-low transition. Using this idea, the completed table is:

	Q2	Q1	Q0
Initial state:	0	0	0
1st clock:	0	0	1
2nd clock:	0	1	0
3rd clock:	0	1	1
4th clock:	1	0	0
5th clock:	1	0	1
6th clock:	1	1	0
7th clock:	1	1	1
8th clock:	0	0	0

By combining the three outputs for each clock pulse, you can see that the circuit is counting in binary from 000 to 111. This translates to a decimal count of from 0 to 7. Notice that the count **rolls over** on the 8th clock pulse. If additional clock pulses were applied, the circuit would continue to count-up and roll over for every eight clock pulses.

Some other observations are worth noting. Look at the maximum count value. For this counter it is binary 111, or decimal 7. On the other hand, there are eight unique count values, from 0 to 7. The number of unique count values that a counter generates is called the **modulus**, or **mod**, of the counter. The modulus of the above counter is, therefore, 8. In the digital world, the counter in Figure 4-36 is sometimes called a mod 8 counter.

Given a straight binary ripple counter such as this, the maximum count value is always one less than the **natural** modulus of the counter. The term natural modulus assumes that the beginning count value of the counter is zero. You can determine the natural modulus and maximum count value of a straight binary counter if you know how many flip-flops are in the circuit. If you designate n as the number of flip-flops in the circuit, then:

$$\text{Natural Modulus} = 2^n$$
$$\text{Maximum Count Value} = 2^n - 1$$

Example 4-7

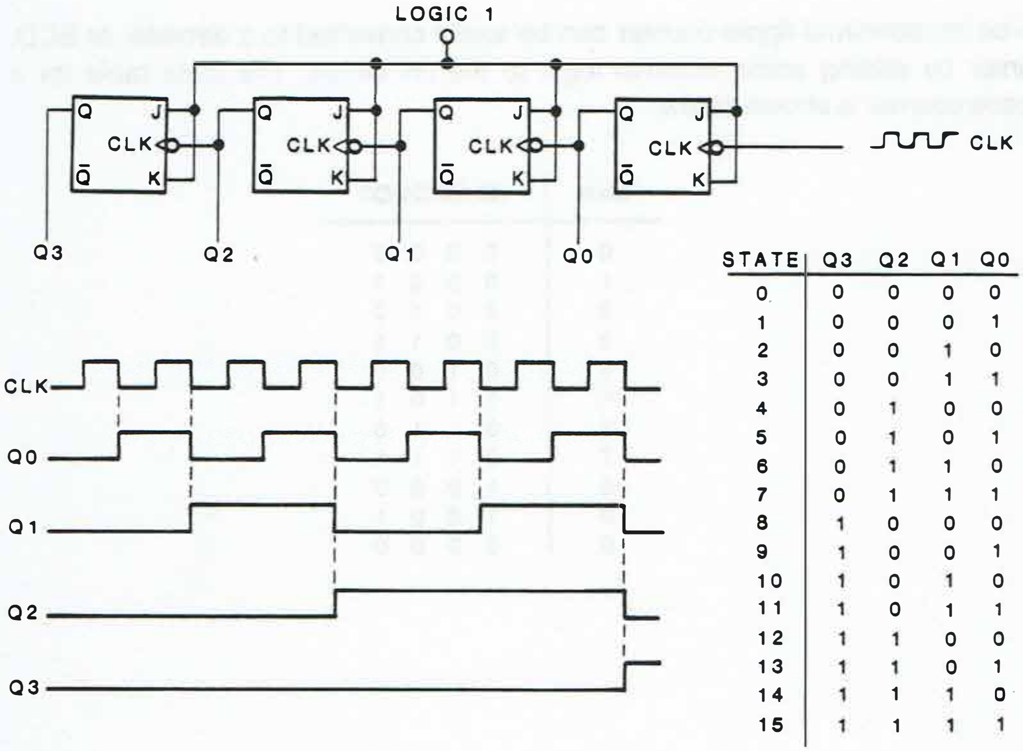
Redesign the 3-bit ripple counter in Figure 4-36 so that it will count down, from 111 to 000.

The simplest way to make a ripple counter count down is to use the $\overline{Q}$ outputs instead of the Q outputs. Recall that the $\overline{Q}$ outputs are the complement of the Q outputs. If you invert the output logic in the above table you get:

	$\overline{Q2}$	$\overline{Q1}$	$\overline{Q0}$
Initial state:	1	1	1
1st clock:	1	1	0
2nd clock:	1	0	1
3rd clock:	1	0	0
4th clock:	0	1	1
5th clock:	0	1	0
6th clock:	0	0	1
7th clock:	0	0	0
8th clock:	1	1	1

As you can see, the $\overline{Q}$ outputs are counting down from 111 to 000, or 7 to 0, while the Q outputs are counting up. Another way to make a down counter is to use the Q outputs, but clock each flip-flop with the $\overline{Q}$ output of the previous flip-flop. Of course the least significant flip-flop is still clocked by the external clock signal. Try it!

You can increase the size, or modulus, of a ripple counter by simply adding additional flip-flop stages. For instance, a 4-bit ripple counter is shown in Figure 4-37, along with its associated truth table. The truth table for a counter is called a **state table**, since it shows the sequential output states of the counter. Notice from the state table in Figure 4-37 that this is a mod 16 counter which will count up from 0000 to 1111, or 0 to 15. This particular mod 16 counter is also called a hexadecimal, or hex, counter for obvious reasons.



STATE	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Figure 4-37

A 4-bit ripple counter.

The timing diagram in Figure 4-37 illustrates another aspect of a counting circuit. A counting circuit can be viewed as a frequency divider. The timing diagram clearly shows this. Since each JK flip-flop divides its clock frequency by two, the counter in Figure 4-37 generates the following frequencies at its four outputs:

Frequency at Q0 = $f_{CLK}/2$
Frequency at Q1 = $f_{CLK}/4$
Frequency at Q2 = $f_{CLK}/8$
Frequency at Q3 = $f_{CLK}/16$

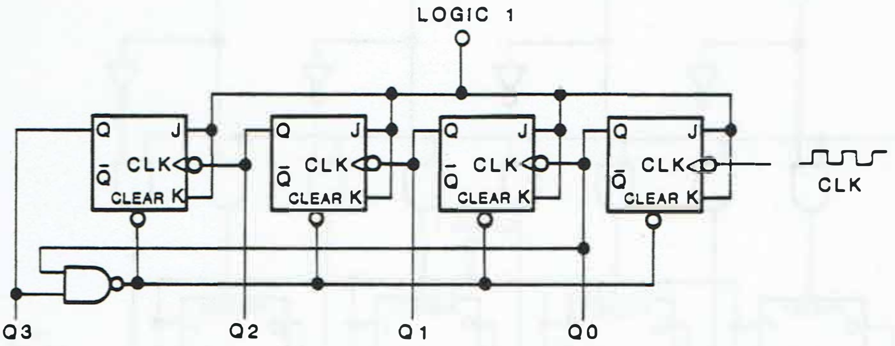
In terms of frequency, the counter in Fig. 4-37 is sometimes called a **divide-by-16** counter.

A Decade, or BCD, Counter

A 4-bit hexadecimal ripple counter can be easily converted to a **decade**, or BCD, counter by adding some external logic to the JK circuit. The state table for a decade counter is shown below.

State	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
0	0	0	0	0

The table shows that after the 9th state, the counter must roll over to 0000, rather than continue to count up. In other words, the counter must **clear** after a count of 1001. Notice that this is the only count for which both Q0 AND Q3 are high. As a result, the simplest way to make a BCD counter is to connect the Q0 and Q3 outputs to a 2-bit AND gate and use the AND gate output to clear the flip-flops. The required alteration is shown in Figure 4-38. Here, a NAND gate is used, since the CLEAR inputs are active low.



STATE	Q
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
0	0000

Figure 4-38
A decade, or BCD, counter.

Presettable Counters

A presettable counter is a counter whose beginning count value can be set anywhere within its counting range. A 4-bit presettable ripple counter is shown in Figure 4-39. Here, the LOAD control line loads the counter with a beginning count value, **D**. When LOAD is high, the AND gates are enabled and a given flip-flop is either preset or cleared. It is preset if the respective data input bit is high and cleared if the data input bit is low.

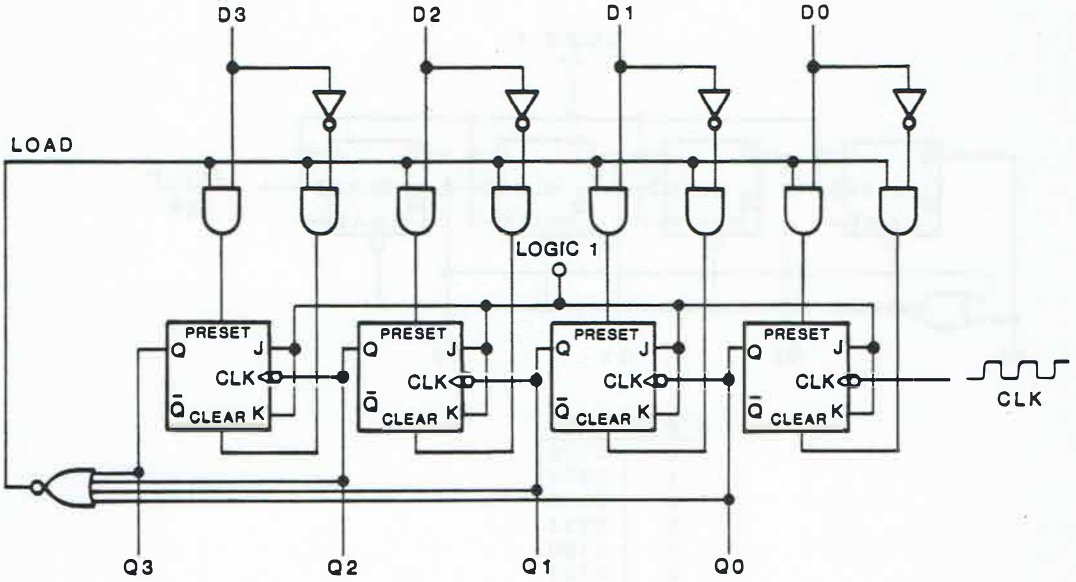


Figure 4-39

A 4-bit presettable ripple counter.

For instance, if **D** = 0100 and LOAD where high, the counter would be loaded with a beginning count value of 0100. Once LOAD is made inactive, or low, the counter begins counting at the preset value, 0100. The counter will count up to its maximum count value, or 1111, then roll over to 0000. However, as soon as it rolls over, the 4-bit NOR gate sees all 0's on its input lines. This causes the NOR gate to generate a logic 1 which activates the LOAD line and returns the counter to the original beginning count value, 0100. In our example, the counter would count from 4 to 15, skipping counts of 0, 1, 2, and 3.

The circuit in Figure 4-39 can be preset with any beginning count value between 0000 and 1111 by simply applying the desired value to the data input lines. This, in turn, determines the modulus of the counter. In the above example, the beginning count value was 0100, or 4. The counter started with this value and counted up to its maximum count value of 1111, or 15. As a result, there were

12 counts from 4 through 15. By definition of modulus, we have made this a mod 12 counter. Likewise, a beginning count value of 0110, or 6, would make a 4-bit ripple counter into a mod 10 counter. In general, the modulus of a counter can be determined as follows:

Modulus = (Max. Count Value − Beginning Count Value) + 1

THE SYNCHRONOUS COUNTER

By now, you should be able to guess where the name **ripple** counter comes from. The term "ripple" comes from the idea that the clocking is "rippled" through the circuit, much like the ripple effect that you get when throwing a stone into a pond. On the other hand, the term "synchronous" means "happening at the same time", according to Webster. Looking at the 3-bit synchronous counter in Figure 4-40 it is clear where this name comes from. (Hint: Look at the clocking.)

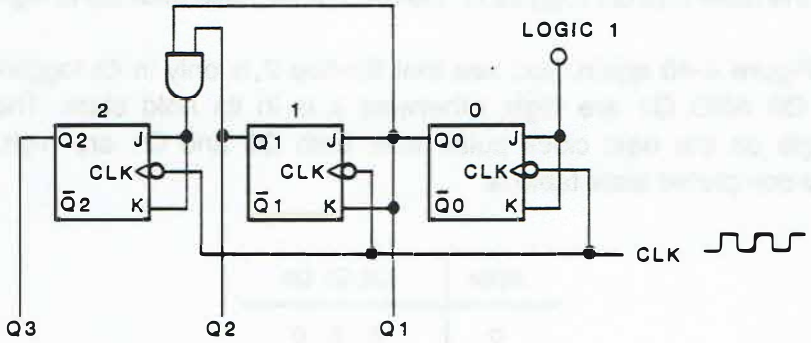


Figure 4-40
A 3-bit synchronous counter.

Flip-flop 0, in Figure 4-40, is hardwired in its toggling state. As a result, its output will toggle with every clock pulse. Now, the only time that Q1 will toggle is when the Q0 output is high, since this applies a logic 1 to both the J and K inputs of flip-flop 1. When Q0 is low, Q1 will not change, since flip-flop 1 is in its hold state. Using this idea, the Q1 output becomes:

State	Q2	Q1	Q0
0		0	0
1		0	1
2		1	0
3		1	1
4		0	0
5		0	1
6		1	0
7		1	1

Notice from the table that Q1 toggles on the next clock pulse after Q0 is high.

Looking at Figure 4-40 again, you see that flip-flop 2 is only in its toggling state when both Q0 AND Q1 are high, otherwise it is in its hold state. Therefore, Q2 will toggle on the next clock pulse after both Q0 and Q1 are high. Using this idea, the completed state table is:

State	Q2	Q1	Q0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Notice that Q2 only toggles when Q0 and Q1 are high.

Like the 3-bit ripple counter described earlier, you see that the synchronous counter performs the same counting function. The size, or modulus, of the synchronous counter can be easily increased by using additional flip-flop stages as shown in Figure 4-41. See if you can trace the logic through this counter and get a state table that counts up from 0000 to 1111.

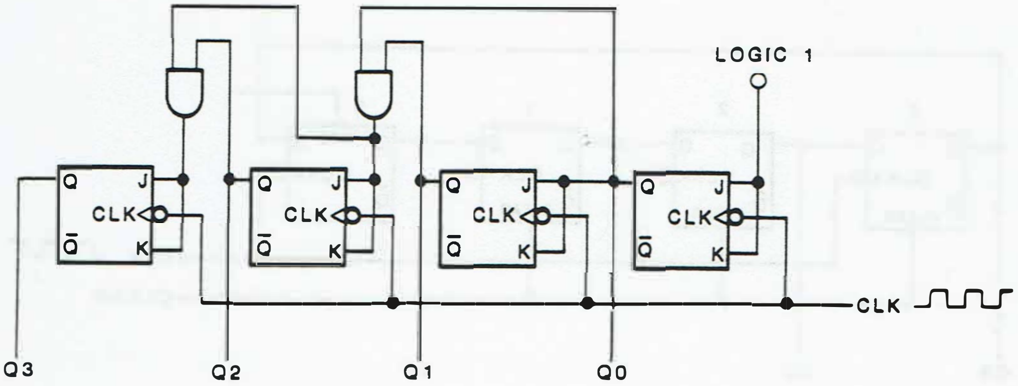


Figure 4-41
A 4-bit synchronous counter.

Example 4-8

Suppose the flip-flops used for the 4-bit counters in Figures 4-37 and 4-41 have a propagation delay time of 20 ns. For each circuit, how long after a negative clock edge hits does it take for the output to roll over from 1111 to 0000?

The flip-flops in the 4-bit ripple counter (Figure 4-37) are clocked sequentially, from one flip-flop to the next. Thus, it takes four propagation delay times, or 4 × 20 nS, for all the flip-flops to toggle.

The flip-flops in a synchronous counter (Figure 4-41) are all clocked at the same time, resulting in at most one propagation delay time, or 20 nS, between counts.

The above example shows you the speed advantage that a synchronous counter has over a ripple counter. However, the price you must pay is in extra logic, resulting in a slightly higher component cost for the synchronous counter.

THE RING COUNTER

The 4-bit ring counter shown in Figure 4-42 uses D flip-flops. First, notice that the CLEAR line presets the least significant flip-flop, 0, and clears the remaining flip-flops. As a result, activating the CLEAR line generates an output of **Q** = 0001. Next, you see that the flip-flops are clocked synchronously and that the output of one flip-flop is applied to the input of the next.

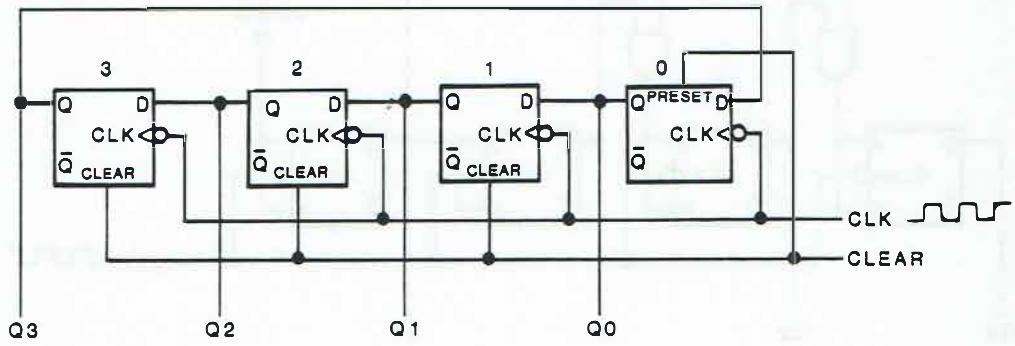


Figure 4-42
A 4-bit ring counter uses D flip-flops.

Assuming the CLEAR line is temporarily activated to preset the counter to 0001, what will happen when it is clocked? After the CLEAR operation, a logic 1 is applied to flip-flop 1, and a logic 0 is applied to the remaining flip-flops. When the first clock hits, flip-flop 1 will set and the remaining flip-flops will reset. This results in an output of **Q** = 0010.

As this point, a logic 1 is applied to flip-flop 2, and a logic 0 to the remaining flip-flops. Thus, when the next clock pulse hits, flip-flop 2 will set and the remaining flip-flops will reset. The resulting output is **Q** = 0100. Notice how the logic 1 is being **rotated** through the output in a counterclockwise direction. This process continues, to generate the following state table:

State	Q3	Q2	Q1	Q0
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

Here, the output is not an arithmetic type of count that you observed with the previous counting circuits. Nevertheless, you could say that the counter is counting in powers of 2. The real usefulness of the ring counter comes from the idea that it sequentially **rotates** a logic 1 to the left, in a counterclockwise direction. This type of operation is called a **rotate left** operation. How would you redesign the circuit in Figure 4-42 to produce a **rotate right** operation?

Example 4-9

A certain industrial process requires that four valves be actuated sequentially at 1 second intervals. Design a digital control circuit that will provide the necessary control logic.

The obvious solution is the 4-bit ring counter shown in Figure 4-43. The sequence is started by clearing the counter. This activates valve 1. After this, each subsequent valve is sequentially activated as a logic 1 is rotated through the counter. A 1 Hz clock is used to generate the required 1 second intervals.

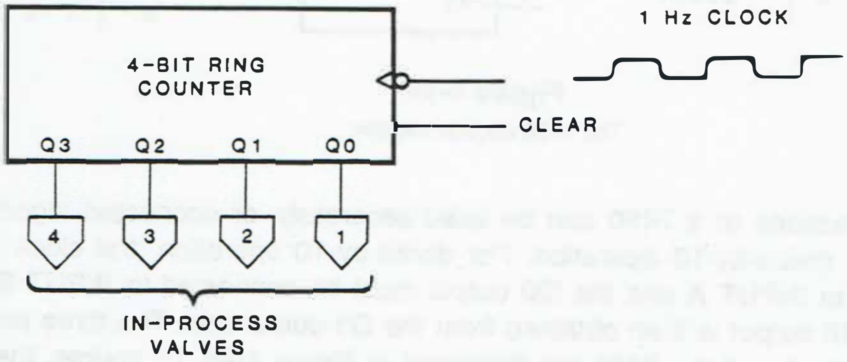


Figure 4-43
Ring counters are used to control sequential operations (Example 4-9).

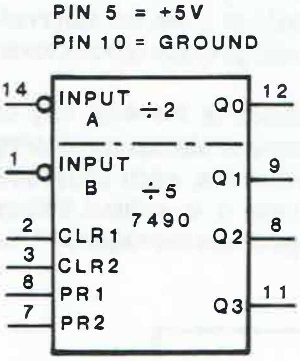
SOME REAL COUNTER DEVICES

The following description is an overview of some of the more popular MSI counting devices readily available on the commercial market. You will find that all of these are variations of the basic ripple counter discussed above.

The 7490 Decade, or BCD, Counter

A functional block diagram for the 7490 is shown in Figure 4-44. It actually consists of two separate counting sections: A and B. Section A is a divide-by-two section consisting of a single JK flip-flop. Its clock input is labeled INPUT A and its output is Q0. Section B is a divide-by-five section consisting of two JKs and an RS flip-flop. Its clock input is labeled INPUT B and its outputs are Q1, Q2, and Q3. The Q1 output divides the INPUT B frequency by two, Q2 divides INPUT B by four, and the last output, Q3, generates the divide-by-five output.

CLR1	CLR2	PR1	PR2	Q3	Q2	Q1	Q0
1	1	0	X	0	0	0	0
1	1	X	0	0	0	0	0
X	X	1	1	1	0	0	1
0	X	0	X	COUNT			
X	0	X	0	COUNT			
0	X	X	0	COUNT			
X	0	0	X	COUNT			



STATE	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Figure 4-44

The 7490 decade counter.

The two sections of a 7490 can be used separately, or connected together to produce a divide-by-10 operation. For divide-by-10 operation, the clock signal is applied to INPUT A and the Q0 output must be connected to INPUT B. The divide-by-10 output is then obtained from the Q3 output line. The three possible operating modes of the 7490 are illustrated in Figure 4-45. Of course, the logic on the combined outputs also generates the BCD count shown in the state table in Figure 4-44, when the counter is operating in the divide-by-10 mode.

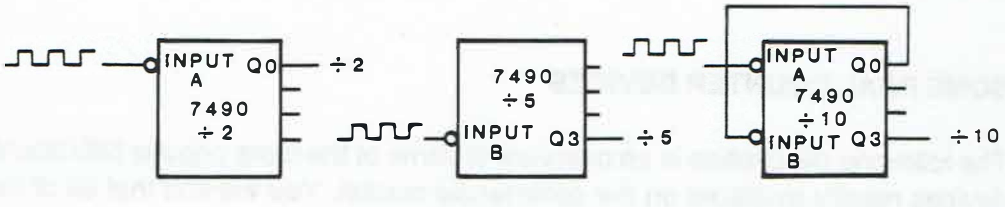


Figure 4-45

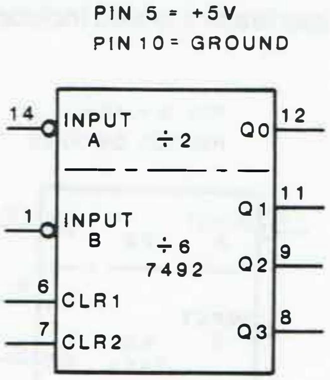
The three operating modes of the 7490.

Notice also from the 7490 diagram in Figure 4-44 that there are two pairs of control lines connected to the PRESET (PR) and CLEAR (CLR) inputs. The logic on these lines allows the counter to be cleared to 0000 or preset to 1001. The associated truth table defines the required logic.

The 7492 Divide-by-12 Counter

The 7492 counter, shown in Figure 4-46, is organized very much like the 7490. It has two separate sections: a divide-by-2 section (A) and a divide-by-6 section (B). The two sections can be connected together to produce a 2 × 6, or divide-by-12, operation. These three operating modes are illustrated in Figure 4-47.

CLR1	CLR2	OPERATION
1	1	CLEAR
0	1	COUNT
1	0	COUNT
0	0	COUNT



STATE	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	1	0	0	0
7	1	0	0	1
8	1	0	1	0
9	1	0	1	1
10	1	1	0	0
11	1	1	0	1

Figure 4-46
The 7492 ÷ 12 counter.

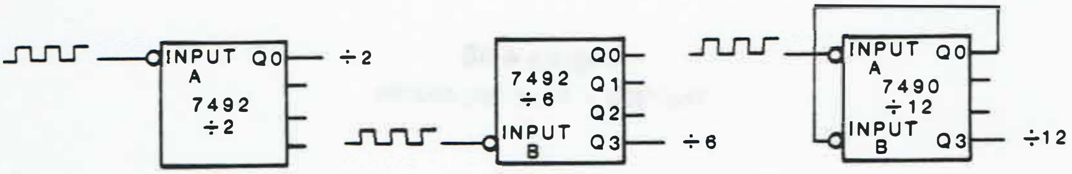


Figure 4-47
The three operating modes of 7492.

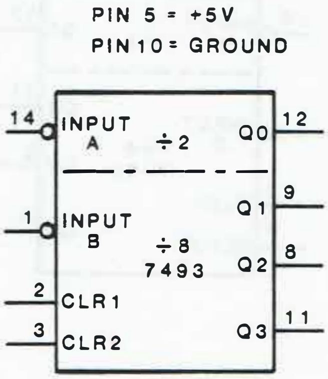
By looking at the 7492 state table, you find that it does **not** count from 0 to 11. Rather, it counts from 0 to 5 then from 8 to 13. For this reason, it is not normally used in binary counting operations. The 7492 is more useful for frequency division applications. Can you think of a "timely" example where a 7492 might be used?

Also, notice from Figure 4-46 that the 7492 includes a clear feature but not a preset feature.

The 7493 Divide-by-16, or Hex, Counter

The 7493 is a straight 4-bit ripple counter as you can see from Figure 4- 48. Again, it consists of two sections: a divide-by-2 section (A) and a divide-by-8 section (B). The two sections can be connected together in the same way as the 7490 and 7492 to generate a divide-by-16 count. In addition, in the divide-by-16 mode, the four outputs provide a 4-bit binary count, from 0000 to 1111, as shown in the state table. A clear feature is also included on this device.

CLR1	CLR2	OPERATION
1	1	CLEAR
0	1	COUNT
1	0	COUNT
0	0	COUNT



STATE	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Figure 4-48
The 7493 ÷ 16, or hex, counter.

Example 4-10

Design a digital circuit to divide a given frequency by 400 using counter ICs.

The three ICs shown in Figure 4-49 will do the job. Notice that one 7490 is connected for divide-by-10 operation and the second 7490 is operating in its divide-by-5 mode. The 7493 is operating in its divide-by-8 mode. If the input frequency is applied to INPUT A of the first 7490, the Q3 output of the 7493 will generate a divide-by-(10 × 5 × 8), or 400, frequency.

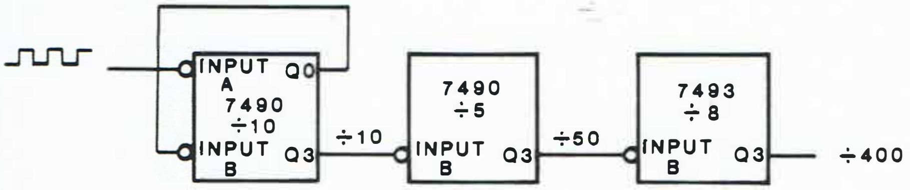


Figure 4-49

A ÷ 400 counter circuit.

Several other popular counters are listed in Table 4-3. When using any of these devices, you will want to consult the respective IC data sheets to determine how they operate.

Table 4-3: Some Common Counter ICs

Device Number	Function
7490	Decade, BCD, Divide-by-10
7492	Divide-by-12
7493	Hex, Divide-by-16
74160	Presettable Decade
74161	Presettable Hex
74190	Up/Down Presettable Decade
74191	Up/Down Presettable Hex
74192	Up/Down Decade
74193	Up/Down 4-bit Binary

Self-Test Review

14. Suppose the initial conditions for an 8-bit shift-left register are as follows:

$$\mathbf{D} = 0000\ 0011$$
$$D_{in} = 0$$

A. Determine the register output for eight consecutive clock pulses.

B. What is the register doing, if the above outputs are interpreted arithmetically? ______________________________

____

____

15. Suppose the initial conditions for an 8-bit shift-right register are as follows:

$$\mathbf{D} = 1000\ 0000$$
$$D_{in} = 0$$

A. Determine the register output for eight consecutive clock pulses.

B. What is the register doing arithmetically? ____________________

____

____

16. How many clock cycles would it take to transmit n-bits of serial data using a shift-right register? ____________________

17. How long would it take to load an 8-bit serial input shift register using a 1 MHz clock signal? ____________________

18. How long would it take to load a parallel input shift register using a 1 MHz clock? ____________________

19. The term **baud rate** is used to describe how many bits per second are being transferred on a serial communications link. A **modem** is a device that is used to transmit and receive serial data over the telephone lines. Suppose that you wish to transmit a 6000 word text over the phone lines using a 300 baud modem. Assume that the average length of each word is 6 characters, including spaces between words (spaces are transmitted as characters). How long would the transmission take if each character must be represented with an 11-bit serial word? ____________________

20. What is the natural modulus and the maximum count value for a 16-bit counter? ______________________

21. An 8-bit counter is preset to the decimal value 125. What is the modulus of the counter? ______________________

22. The circuit in Figure 4-50 is a divide-by-________ counter.

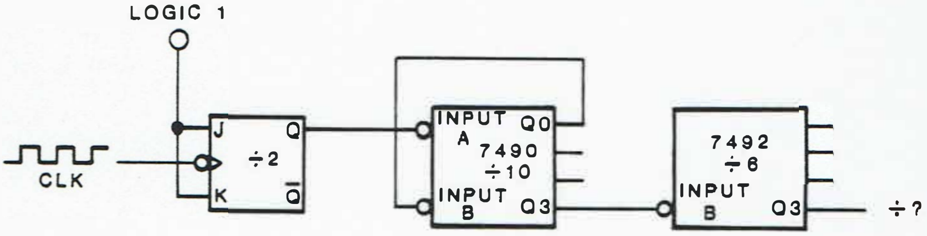


Figure 4-50

The counter circuit for Question 22.

23. Design a divide-by-60 counter using the ICs described in this section.

24. Using a 60 Hz clock signal, design a digital counting circuit that will generate a signal every second, minute, and hour. (Hint: use the divide-by-60 counter that you designed in question 23.)

25. How could you make the digital clock circuit in question 24 generate a signal to indicate AM and PM? ______________________

____

26. Design a 4-bit rotate-right ring counter.

27. Suppose that a rotate-left ring counter were preset with the value **Q** = 1110.

A. Develop the state table for four consecutive clock pulses.

B. How could these outputs be obtained from the ring counter in Figure 4-42? ______________________

____

28. Look at Figure 4-51. This figure illustrates how buffer registers are used with bus systems. Notice how the **control register** generates the control signals for the other registers. The output of the control register is a 6-bit word defined as follows:

C = L3 E3 L2 E2 L1 E1

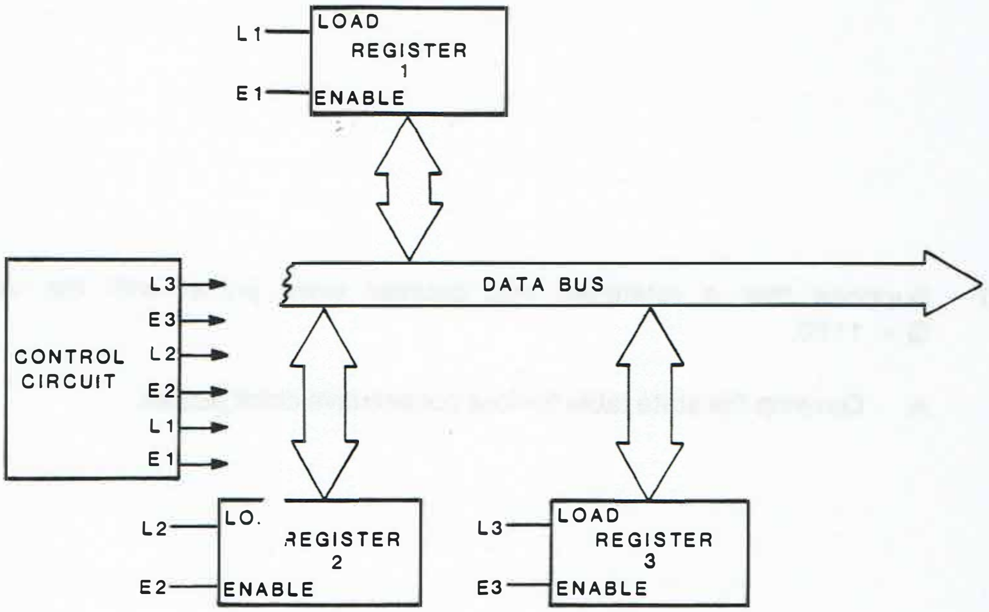


Figure 4-51

Registers and bus systems, Question 28.

A. What will happen when **C** = 100000? ______________________

B. What will happen when **C** = 001010? ______________________

C. What will happen when **C** = 000100? ______________________

D. What will happen when **C** = 100001? ______________________

Answers

14A. 1st clock: 0000 0110
2nd clock: 0000 1100
3rd clock: 0001 1000
4th clock: 0011 0000
5th clock: 0110 0000
6th clock: 1100 0000
7th clock: 1000 0000
8th clock: 0000 0000

14B. The output is multiplied by 2 for each clock pulse. This can be determined if you convert the above binary outputs to decimal and look at them as an arithmetic sequence. Notice, however, that after the 6th clock pulse, the output has no meaning since the more significant logic 1 bits are dropped.

15A. 1st clock: 0100 0000
2nd clock: 0010 0000
3rd clock: 0001 0000
4th clock: 0000 1000
5th clock: 0000 0100
6th clock: 0000 0010
7th clock: 0000 0001
8th clock: 0000 0000

15B. Looking at the output sequence arithmetically, the circuit divides the previous output by two for each consecutive clock pulse.

16. It would require n-clock pulses. The first clock pulse would transmit the least significant bit by loading the parallel word into the register. Then, n-1 clock pulses are required to transmit the remaining bits.

17. It would take 8 μs. The period of a 1 MHz clock signal is 1/1 MHz, or 1 microsecond (1 μS). Since eight clock pulses are required to load an 8-bit serial input shift register, it would take 8 × 1 μS, or 8μS to load the register.

18. It would only take one clock cycle, or 1 μS, to load an 8-bit parallel input shift register, since all the input bits are loaded at the same time.

19. Solution follows.

20. The natural modulus for a 16-bit counter is: 2^{16}, or 65,536. The maximum count for a 16-bit counter is: $2^{16} - 1$, 65,535.

Answers (continued)

21. 131

22. 120

23. A divide-by-60 counter can be designed two different ways using a 7490 and a 7492. First, the 7490 can be used in its divide-by-10 mode, followed by the 7492 in its divide-by-6 mode. Second, 7490 can be connected in its divide-by-5 mode and the 7492 in its divide-by-12 mode.

24. See Figure 5-52. The signals generated by the circuit can then be used to clock counters which count in seconds, minutes, and hours. This is the principle behind a digital clock. The 60 Hz line frequency is obtained from the standard 110-volt household line signal.

25. A 7492 divide-by-12 counter could be added to the circuit in Figure 4-52 to generate an AM/PM signal.

26. Reverse the output lines of the rotate-left ring counter in Figure 4-42. To do this, Q0 becomes Q3, Q1 becomes Q2, etc.

27A.

State	Output
0	1110
1	1101
2	1011
3	0111
4	1110

27B. By using the $\overline{Q}$ outputs instead of the Q outputs.

28A. Register 3 is loaded with the current value on the data bus.

28B. Registers 1 and 2 are loaded with the current value on the data bus.

28C. The contents of register 2 are placed on the data bus.

28D. The contents of register 1 are transferred to register 3.

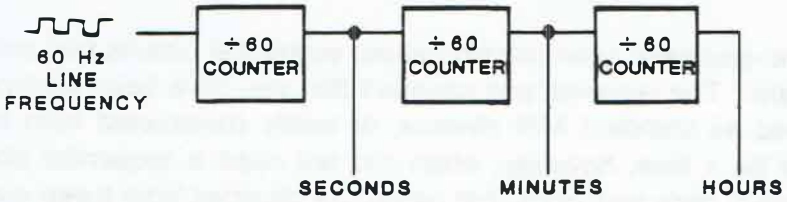


Figure 4-52

A counter circuit that will generate pulses for a digital clock (answer to question 24).

Solution to Questions 19 and 21.

19. If there are an average of six characters per word including spaces, a 6000 word text has approximately 6 × 6000, or 36000 characters. If each character is represented by an 11-bit word, then 11 × 36000, or 396000, bits must be transmitted. Now, the baud rate of the modem is 300. Thus, the modem transmits 300 bits-per-second, which means that each bit takes 1/300 seconds to transmit. This is called the **bit time period**, since it is the amount of time occupied by one bit. Consequently, the total transmission time must be 396000/300 seconds, or 1320 seconds. This translates to 22 minutes to transmit the entire text.

21. The maximum count value for an 8-bit counter is: $2^8 - 1$, or 255. If the counter is preset to the value 125, its modulus is:

$$\begin{aligned}\text{Modulus} &= (\text{Max. Count Value} - \text{Beginning Count Value}) + 1\\ &= (255 - 125) + 1\\ &= 131\end{aligned}$$

Before you read the next section, go to Unit 8 and perform Experiments 11 and 12.

DESIGNING SEQUENTIAL CIRCUITS

Up to this point, you have been learning about sequential circuits that are more or less "standard." The registers and counters that you have been working with can be obtained as standard MSI devices, or easily constructed from flip-flop ICs. There will be a time, however, when you will need a sequential circuit to generate a special code sequence that cannot be obtained from these standard registers and counters. When this time comes, you will need to use a general purpose design procedure that will allow you to satisfy any non-standard code applications. In this section, you will learn such a design procedure. I call it the "sequential circuit design algorithm."

The Sequential Circuit Design Algorithm

Let's review, for a minute, the basic ideas behind a sequential circuit. Recall that a sequential circuit is one that must operate with a sequence of logic operations over a given period of time. A sequential circuit, like a counter, may be designed to generate a code sequence to control the operation of other circuits in a digital system. On the other hand, a sequential circuit, like a shift-register, may be required to process a given sequence of logic signals. Furthermore, some applications require a sequential circuit to detect a certain sequence of events and respond by generating control signals to other circuits.

Regardless of the application, all sequential circuits have one thing in common: memory. As you are now aware, flip-flops provide the memory elements within a sequential circuit. In addition to flip-flop memory, many sequential circuits require combinational logic to process input and output signals. This is especially true of sequential circuits that are designed to handle unique codes and sequences. The black-box diagram in Figure 4-53 summarizes these major features of a sequential circuit.

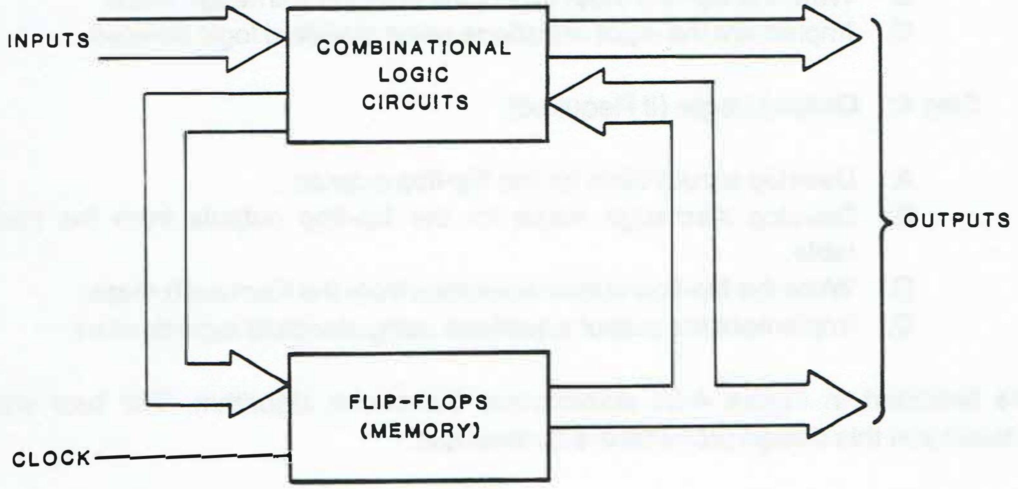


Figure 4-53
A black-box sequential logic circuit.

The Algorithm

The remainder of this unit is devoted to teaching you the techniques required to successfully design special purpose sequential logic circuits. The following design algorithm provides you with the required step-by-step design procedure:

Step 1: **Problem Definition**

A. Develop a design concept using the black-box method.
B. Define all inputs, outputs, and I/O logic from the application task and design concept.
C. Decide how many and what type of flip-flops to use.

Step 2: **State Table**

A. Develop a state table from the problem definition.

Step 3: **Input Logic**

A. Develop Karnaugh maps for the flip-flop inputs from the state table.
B. Write the flip-flop input equations from the Karnaugh maps.
C. Implement the input equations using standard logic devices.

Step 4: **Output Logic** (If Required)

A. Develop a truth table for the flip-flop outputs.
B. Develop Karnaugh maps for the flip-flop outputs from the truth table.
C. Write the flip-flop output equations from the Karnaugh maps.
D. Implement the output equations using standard logic devices.

The flowchart in Figure 4-54 summarizes the above algorithm. The best way to teach you this design procedure is by example.

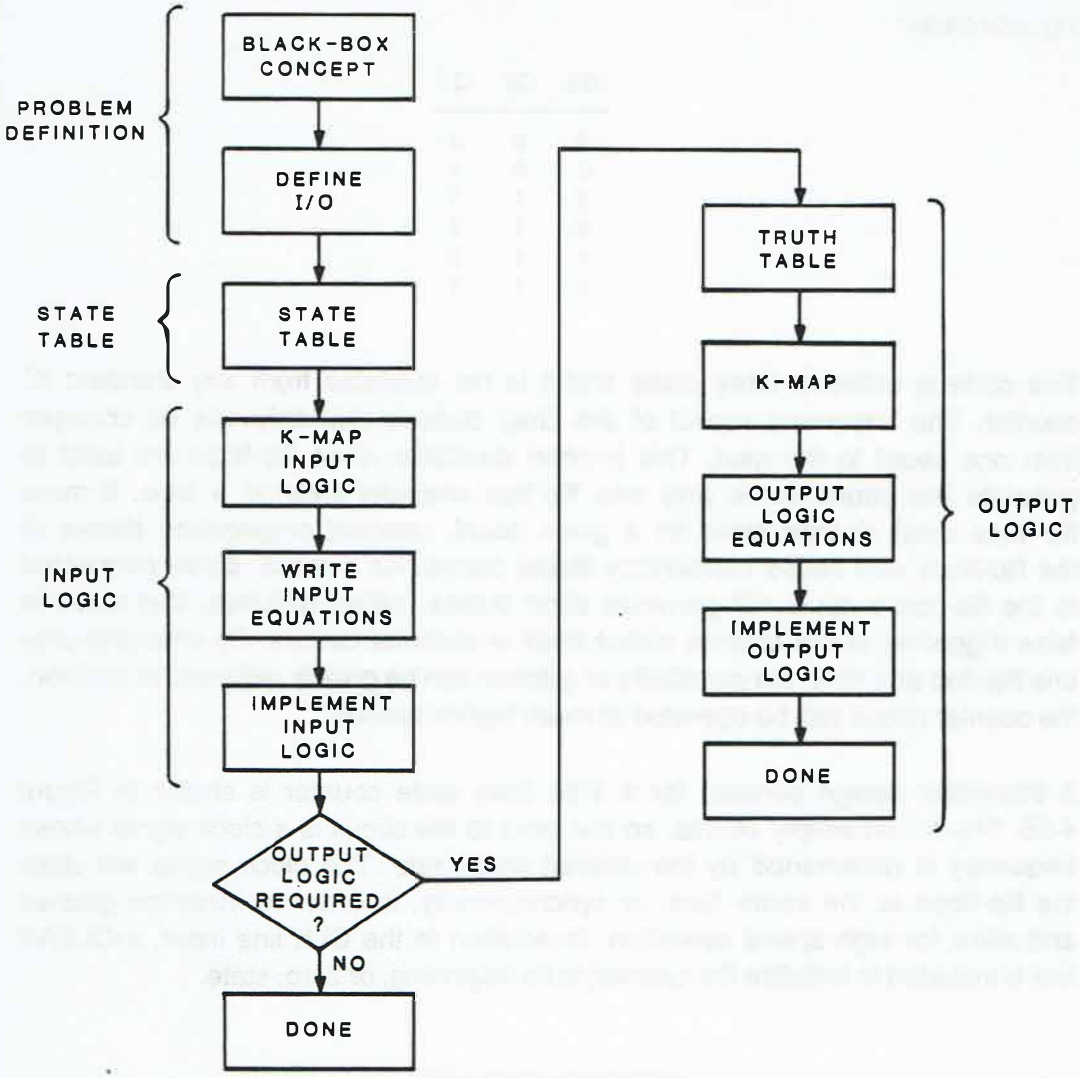


Figure 4-54

A flowchart summary of the sequential circuit design algorithm.

PROBLEM DEFINITION

Suppose you wish to design a sequential logic circuit that will generate the following 3-bit code:

Q3	Q2	Q1
0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1

This code is called a **Gray code** and it is not available from any standard IC counter. The important aspect of the Gray code is that only one bit changes from one count to the next. This is often desirable when flip-flops are used to generate the count, since only one flip-flop changes state at a time. If more flip-flops must change state for a given count, unequal propagation delays of the flip-flops can cause momentary illegal states. As a result, gates connected to the flip-flop outputs will generate short pulses, called **glitches**, that result in false triggering of the counter circuit itself or external circuits. By changing only one flip-flop at a time, the possibility of glitches can be greatly reduced. In addition, the counter circuit can be operated at much higher speeds.

A black-box design concept for a 3-bit Gray code counter is shown in Figure 4-55. The circuit simply counts, so the input to the circuit is a clock signal whose frequency is determined by the desired count rate. The clock signal will clock the flip-flops at the same time, or synchronously, in order to minimize glitches and allow for high speed operation. In addition to the CLK line input, a CLEAR line is included to initialize the counter to its beginning, or zero, state.

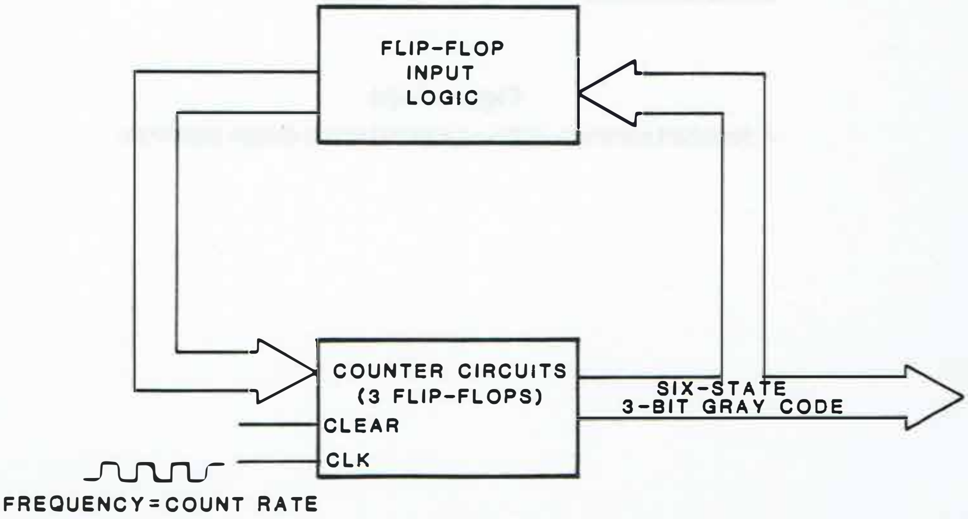


Figure 4-55

A black-box design concept for a 3-bit Gray code counter.

The counter output must be the 3-bit Gray code sequence. However, observe that only six states are included in the sequence. The 101 and 100 logic combinations are not included and will be designated as illegal states. Actually, the Gray code does go beyond six states. However, I have chosen only to use the first six states for example purposes.

The black-box concept in Figure 4-55 shows that combinational feedback logic must be designed to sample the counter outputs and generate the necessary flip-flop input logic to prepare the circuit for the next count. No additional output logic is required for this circuit, since the 3-bit Gray code will be obtained directly from the flip-flop outputs.

The next task is to decide how many and what type of flip-flops must be used for the circuit. Since a 3-bit code must be generated, it is obvious that three flip-flops must be used, one for each bit. You can use either D or JK flip-flops, however JKs often result in fewer external logic gates. I will develop both a D and JK flip-flop design for this example so that you will learn both design techniques.

THE STATE TABLE

The state table for a sequential logic design must show all the legal and illegal output states of the circuit. In addition, the state table must show both the **t** and **t+1** states. The t states show the flip-flop outputs before the arrival of a clock pulse, and the t+1 states show the flip-flop outputs after the clock pulse hits. The required state table for the 3-bit Gray code counter follows:

State	t Q2 Q1 Q0	t+1 Q2 Q1 Q0
0	0 0 0	0 0 1
1	0 0 1	0 1 1
2	0 1 1	0 1 0
3	0 1 0	1 1 0
4	1 1 0	1 1 1
5	1 1 1	0 0 0
— — —	— — — —	— — — —
Illegal	1 0 1 1 0 0	Don't Care

The last two states, 101 and 100, are designated as "illegal" states which result in "don't care" outputs, since the problem definition dictates a six-state Gray code counter.

THE INPUT LOGIC

Input logic must now be designed to sample the circuit outputs and apply the necessary logic to the flip-flop input lines. Here is where the procedure differs between D and JK flip-flop circuits. First, you will see how to design the input logic for D flip-flops. Then, you will see how to design the input logic for JK flip-flops.

The D Flip-Flop Design

Recall that a D flip-flop transfers its input logic to its output when clocked. Consequently, a logic 1 must be applied to a given D flip-flop input when its next, or t + 1, state must be high. Using this idea, locate the t states where a given output is high in the next, or t + 1, state. Notice, for example, that Q2 must be high after t states 3 and 4. Likewise, Q1 must be high after states 1, 2, 3, and 4. Finally, Q0 must be high after states 0, 1, and 4.

The next step is to construct a K-map grid for **each** flip-flop **input**. The grid is constructed using the three outputs (Q2, Q1, and Q0) as the grid variables, as shown in Figure 4-56a. The two most significant output bits are listed along the column, and the least significant output is listed along the row. A separate K-map grid must be made for each flip-flop input (D2, D1, and D0) as shown in Figures 4-56b, c, and d.

First, the two illegal state conditions, 101 and 100, are plotted as don't cares, since they should not occur. Then, respective plots are made for each t state where the t + 1 state is a logic 1. For example, the state table dictates that Q2 must be high after states 3 and 4. At these states, the t-state logic is 010 and 110, respectively. Thus, plots are made for each of these logic combinations on the D2 K-map, Figure 4-56b.

For Q1, states 1, 2, 3, and 4 are of interest. At these states, the t-state logic is 001, 011, 010, and 110, respectively. Consequently, plots are made on the D1 K-map, Figure 4-56c, for each of these logic combinations. Likewise, plots are made for t-states 000, 001, and 110 on the D0 K-map in Figure 4-56d.

Once each K-map has been plotted, the plots are grouped and the groups are read as you have done before. The result is a logic equation for each flip-flop input in terms of the flip-flop outputs. In this example, the K-maps generate the following input logic equations:

$$D2 = Q1\,\overline{Q0}$$
$$D1 = Q1\,\overline{Q0} + \overline{Q2}\,Q0$$
$$D0 = \overline{Q1} + Q2\,\overline{Q0}$$

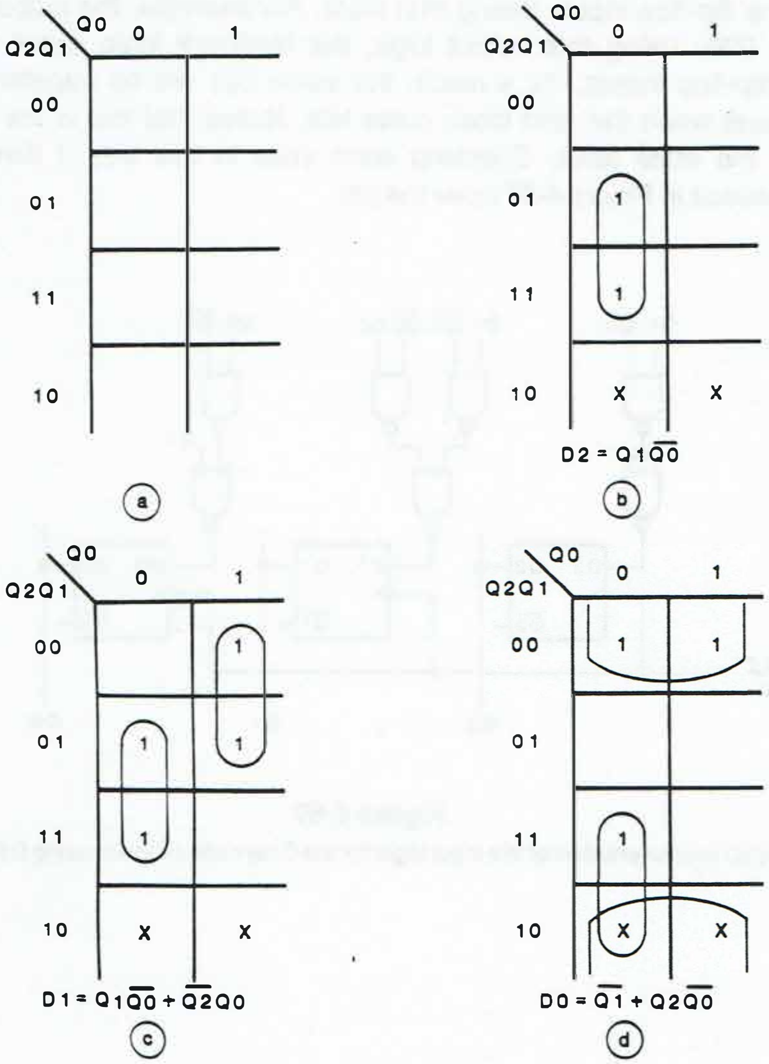

Q0
Q2Q1
0
1
00
01
11
10
a
1
1
X
X
D2 = Q1$\overline{Q0}$
b
1
1
1
1
1
X
X
D1 = Q1$\overline{Q0}$ + $\overline{Q2}$Q0
c
1
1
1
X
X
D0 = $\overline{Q1}$ + Q2$\overline{Q0}$
d

Figure 4-56

The K-maps for the D flip-flop design.

The final step is to use these equations to implement the feedback logic as shown in Figure 4-57. Here, the AND/OR logic in the above equations has been converted to NAND logic. No inverters are required for the feedback logic circuit inputs, since all the $\overline{Q}$ outputs are available from the flip-flops.

To check the circuit, start at a given output state and determine the logic being applied to the flip-flop inputs during that state. For example, the outputs at state 0 are **Q** = 000. Using this output logic, the feedback logic circuit generates 001 to the flip-flop inputs. As a result, the value 001 will be transferred to the flip-flop outputs when the next clock pulse hits. Notice that this is the next state required by the state table. Checking each state in this way, I think you will find that the circuit in Figure 4-57 does the job.

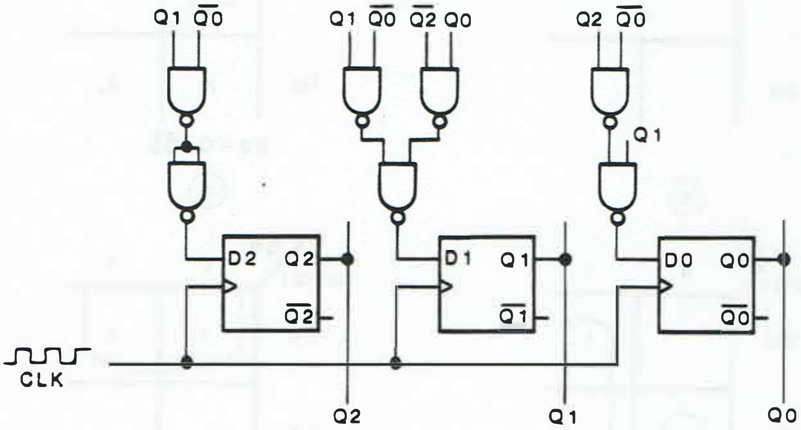


Figure 4-57

The NAND implementation of the input logic for the Gray code problem using D Flip-flops.

The JK Flip-Flop Design

The JK flip-flop design procedure is a little trickier than the D flip-flop design, but often results in a simpler combinational logic circuit. Once a state table is developed, you must do a K-map for each J and K input of each flip-flop. Our circuit requires three flip-flops which result in six K-maps as shown in Figure 4-58.

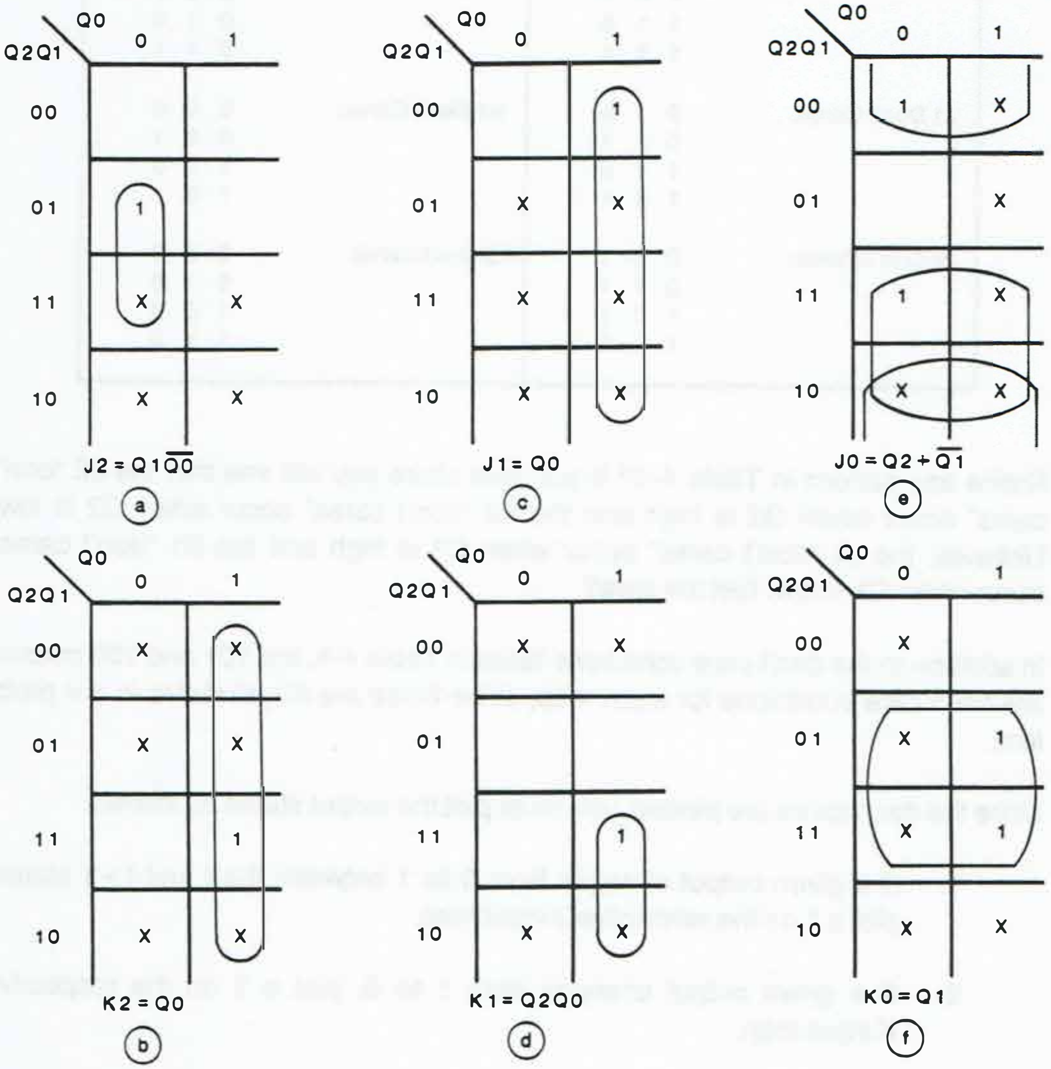


Figure 4-58
The K-maps for the JK flip-flop design.

Notice the extensive use of don't cares in the JK maps. These don't cares **always** result from the action of a JK flip-flop and can be entered immediately on the K-map grid. Table 4-4 shows you the output combinations that will produce don't care conditions for both the J and K inputs.

Table 4-4: J and K Don't Care Conditions

	Q2 Q1 Q0		**Q2 Q1 Q0**
J2 Don't Cares:	1 0 0 1 0 1 1 1 0 1 1 1	K2 Don't Cares:	0 0 0 0 0 1 0 1 0 0 1 1
J1 Don't Cares:	0 1 0 0 1 1 1 1 0 1 1 1	K1 Don't Cares:	0 0 0 0 0 1 1 0 0 1 0 1
J0 Don't Cares:	0 0 1 0 1 1 1 0 1 1 1 1	K0 Don't Cares:	0 0 0 0 1 0 1 0 0 1 1 0

Notice any pattern in Table 4-4? If you look close you will see that the J2 "don't cares" occur when Q2 is high and the K2 "don't cares" occur when Q2 is low. Likewise, the J1 "don't cares" occur when Q1 is high and the K1 "don't cares" occur when Q1 is low. Get the idea?

In addition to the don't care conditions listed in Table 4-4, the 101 and 100 outputs are don't care conditions for each map, since these are illegal states in our problem.

Once the don't cares are plotted, you must plot the output states as follows:

1. If a given output changes from **0 to 1** between the t and t+1 states, plot a 1 on the respective J input map.
2. If a given output changes from **1 to 0**, plot a 1 on the respective K input map.

For instance, the only plot on the J2 map, Figure 4-58a, is for an output of 010, since this is the only place in the state table where Q2 changes from 0 to 1 between the t and t+1 states. The only plot on the K2 map, Figure 4-58b, is for an output of 111, since this is where Q2 must change from 1 to 0.

In the same way, a plot is made on the J1 map, Figure 4-58c, at 001, since this is the only place where Q1 changes from 0 to 1 between the t and t+1 states. Likewise, a plot is made on the K1 map, Figure 4-58d, at 111, since this is where Q1 changes from 1 to 0.

Finally, the J0 plots, Figure 4-58e, are made where Q0 changes from 0 to 1, and the K0 plots, Figure 4-58f, are made where Q0 changes from 1 to 0.

The next thing you must do is to group and read the K-maps. This results in the following JK input equations:

$$\begin{aligned} J2 &= Q1\,\overline{Q0} \\ K2 &= Q0 \\ J1 &= Q0 \\ K1 &= Q2\,Q0 \\ J0 &= Q2 + \overline{Q1} \\ K0 &= Q1 \end{aligned}$$

The final step is to implement the equations and check-out the circuit. Figure 4-59 shows how the AND/OR equations have been implemented with NAND gates. To check the design, determine the J and K input logic generated by the feedback circuit for a given output state. For example, the feedback circuit generates the following J and K input logic when the counter output is 001 in state 1:

$$\begin{aligned} J2\,K2 &= 0\,1 \\ J1\,K1 &= 1\,0 \\ J0\,K0 &= 1\,0 \end{aligned}$$

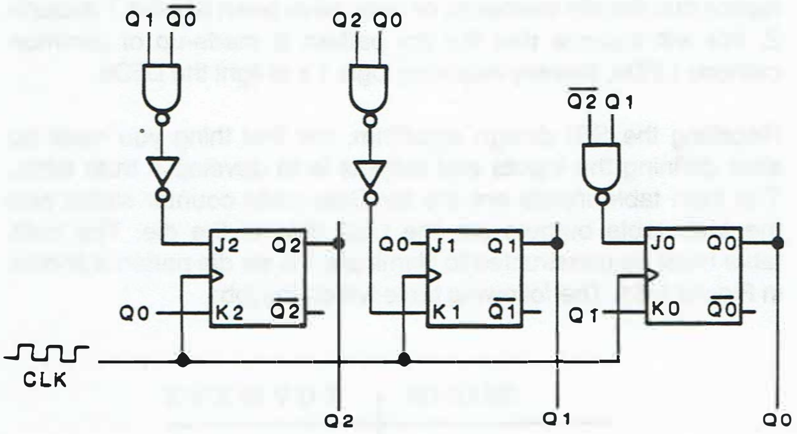


Figure 4-59

The NAND implementation of the input logic for the Gray code problem using JK flip-flops.

With this JK input logic and an active clock pulse, flip-flop 2 will clear, flip-flop 1 will set, and flip-flop 0 will set. This results in the correct t+1 state of 011, according to the state table. You might want to check the remaining states just for practice with JK flip-flops.

OUTPUT LOGIC

The only time you must design output logic is when the code generated by the sequential circuit is not the same as that required by the external application circuit. When this is the case, a code converter circuit must be designed to convert the sequential circuit code to a code that can be used by the application circuit. Thus, the output logic design procedure reduces to the SSI and MSI design techniques that you studied in previous units.

The Gray code circuit just completed does not require any output logic, since the problem definition did not call for it. However, lets extend this design to an example problem that does require output logic.

Example 4-11

Suppose the Gray code generating circuit developed earlier is used to drive an electronic die. Design an output circuit that will convert the six-state Gray code counter output to a die code. Assume the die elements are illuminated using common cathode LEDs.

The idea is to design a code converter circuit that will act as an **interface** between the counter output and the electronic die inputs. A black-box design concept is shown in Figure 4-60. Notice that the die elements, or dots, have been labeled T through Z. We will assume that the dot pattern is made-up of common cathode LEDs, thereby requiring logic 1's to light the LEDs.

Recalling the SSI design algorithm, the first thing you must do after defining the inputs and outputs is to develop a truth table. The truth table inputs are the six Gray code counter states and the truth table outputs are the LED dots of the die. The truth table must be constructed to illuminate the six die patterns shown in Figure 4-61. The following table will do the job:

	Q2	Q1	Q0	T	U	V	W	X	Y	Z
	0	0	0	0	0	0	1	0	0	0
	0	0	1	0	0	1	0	1	0	0
	0	1	1	0	0	1	1	1	0	0
	0	1	0	1	0	1	0	1	0	1
	1	1	0	1	0	1	1	1	0	1
	1	1	1	1	1	1	0	1	1	1
Illegal	1	0	1	Don't Cares						
	1	0	0							

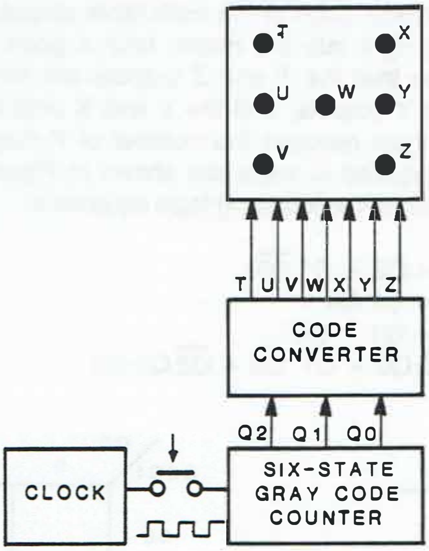


Figure 4-60

The black-box design concept for the die problem in Example 4-11.

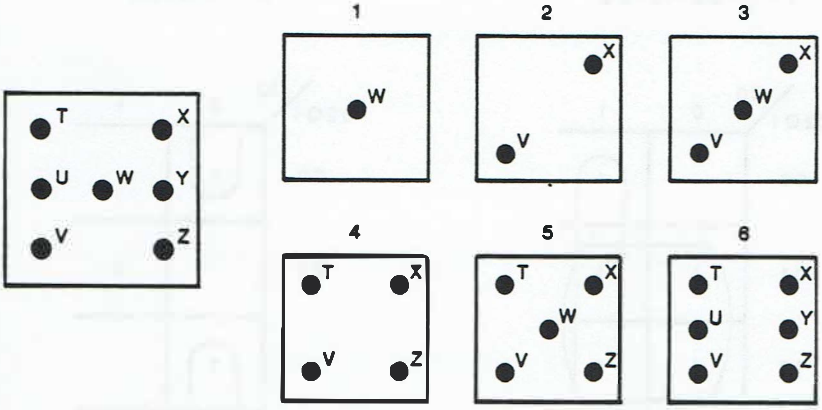


Figure 4-61

The six output states required for the die problem.

Don't forget the don't cares, since they will help you during the K-mapping process.

The next step is to K-map each of the truth table outputs. However, before jumping right into the maps, take a good look at the truth table. Notice that the T and Z outputs are the same, as well as the U and Y outputs, and the V and X outputs. This little observation has just reduced the number of K-maps from seven to four. The required K-maps are shown in Figure 4-62. These mappings generate the following logic equations:

$$
\begin{aligned}
T &= Z = Q2 + Q1\,\overline{Q0} \\
U &= Y = Q2\,Q0 \\
V &= X = Q1 + Q0 \\
W &= Q2\,\overline{Q0} + \overline{Q1}\;\overline{Q0} + \overline{Q2}\,Q1\,Q0
\end{aligned}
$$

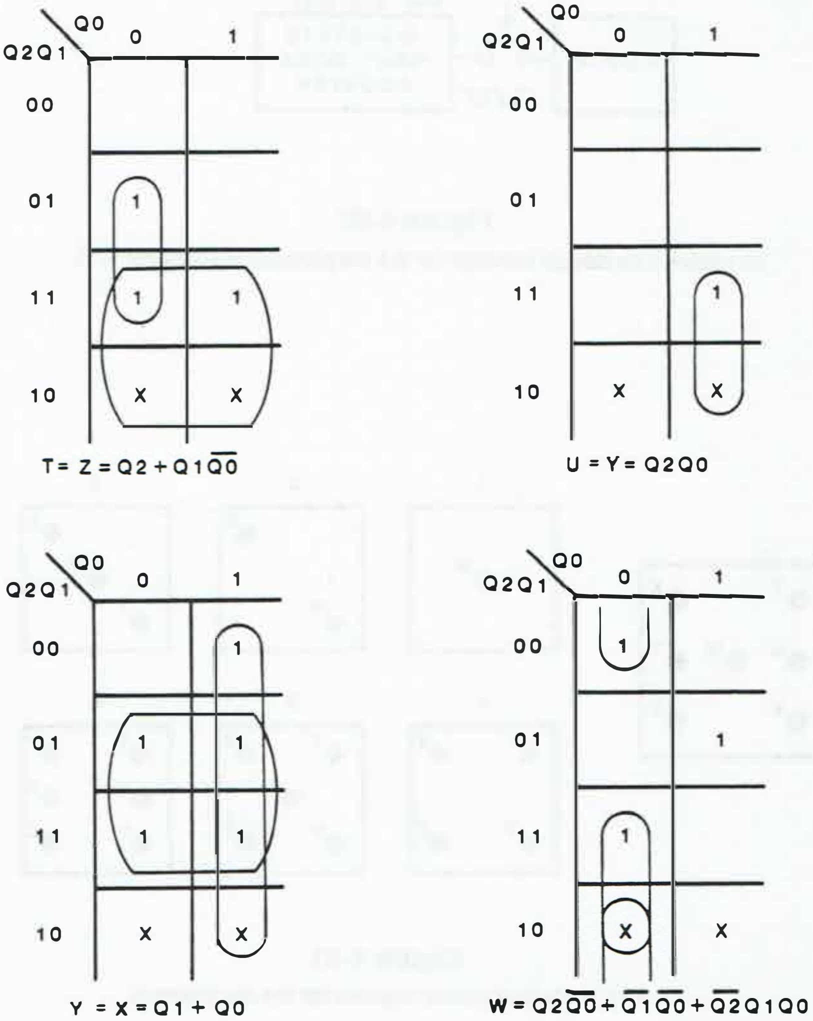

Q0
Q2Q1
0
1
00
01
11
10
1
X
T = Z = Q2 + Q1$\overline{Q0}$
U = Y = Q2Q0
Y = X = Q1 + Q0
W = Q2$\overline{Q0}$ + $\overline{Q1}\,\overline{Q0}$ + $\overline{Q2}$Q1Q0

Figure 4-62

The J and K K-maps for Example 4-11.

These equations are implemented using NAND gates in Figure 4-63. The circuit could be connected to the outputs of either the D flip-flop design in Figure 4-57 or the JK flip-flop design in Figure 4-59. Of course, the converter circuit outputs, T through Z, must be connected to their respective die inputs.

Now, you are ready to "throw" the die. But, you can't actually throw it without destroying your circuit. Rather, a pushbutton switch can be used as shown back in Figure 4-60 to interrupt the counter clock signal. You can use a high frequency clock signal to introduce randomness into the die output. Recall that a Gray code counter can operate at very high speeds without generating glitches. When the pushbutton is pressed, the sequential circuit will count at a very high rate until the pushbutton is released. The counter output at the time the pushbutton is released is converted and displayed by the die. You might want to try building this circuit — it's a fun rainy-day project.

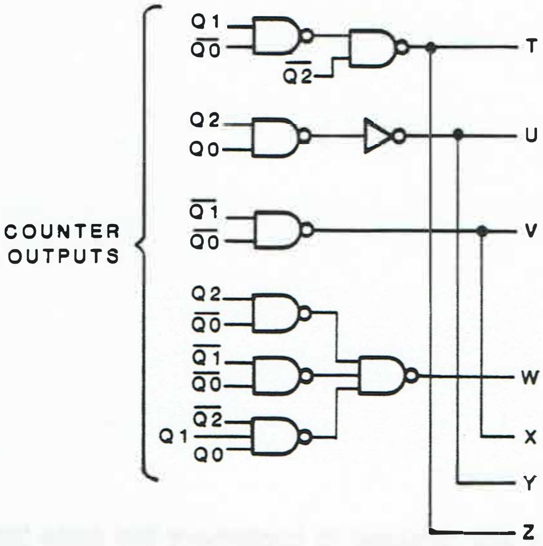


Figure 4-63

The code converter output circuit for Example 4-11.

Self-Test Review

29. List the four major steps of the sequential circuit design algorithm.

A. ______________________________

B. ______________________________

C. ______________________________

D. ______________________________

30. Develop a state table for a 3-bit XS3 counter.

31. How many flip-flops are required to implement the state table in question 30? ____________

32. Design a D flip-flop circuit that will implement the XS3 state table in question 30.

33. Design a JK flip-flop circuit that will implement the XS3 state table in question 30.

34. Explain when an output logic circuit must be designed for a sequential circuit. ______________________________

Answers

29A. Problem Definition

29B. State Table

29C. Input Logic

29D. Output Logic (If Required)

30. Solution follows.

31. Three

32. The required K-maps and resulting circuit are shown in Figure 4-64 on Page 4-90.

33. The required K-maps and resulting circuit are shown in Figure 4-65 on Page 4-91.

34. An output logic circuit must be designed for a sequential circuit when the circuit output code is not compatible with the code required by the application circuit.

Solution to Question 30.

30. Recall that XS3 code is determined by adding 3 to the straight binary code. The resulting 3-bit XS3 state table is:

	t			t+1		
State	Q2	Q1	Q0	Q2	Q1	Q0
0	0	1	1	1	0	0
1	1	0	0	1	0	1
2	1	0	1	1	1	0
3	1	1	0	1	1	1
4	1	1	1	0	1	1
Illegal	0	0	0	Don't Cares		
	0	0	1			
	0	1	0			

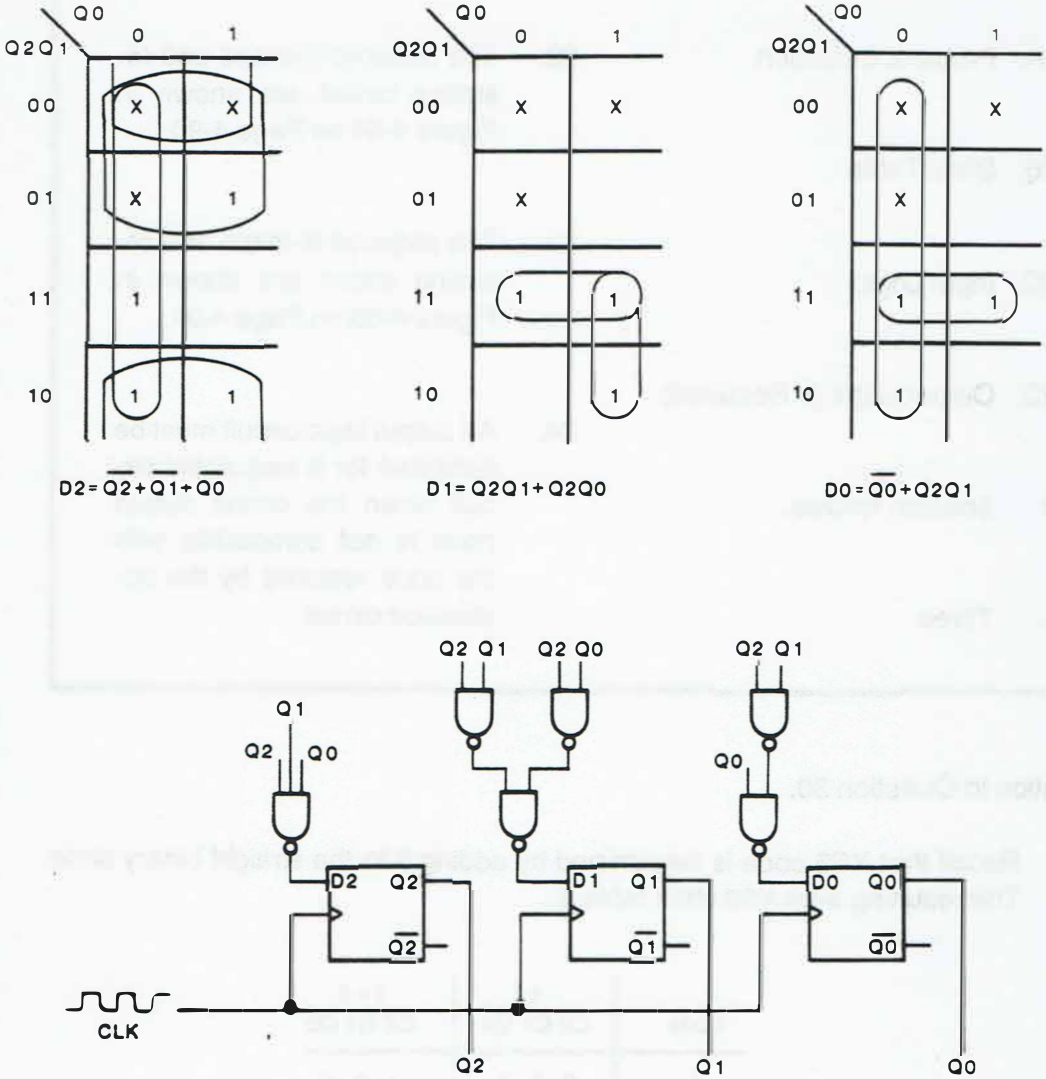


Figure 4-64

The D flip-flop design for a 3-bit XS3 counter (Question 32).

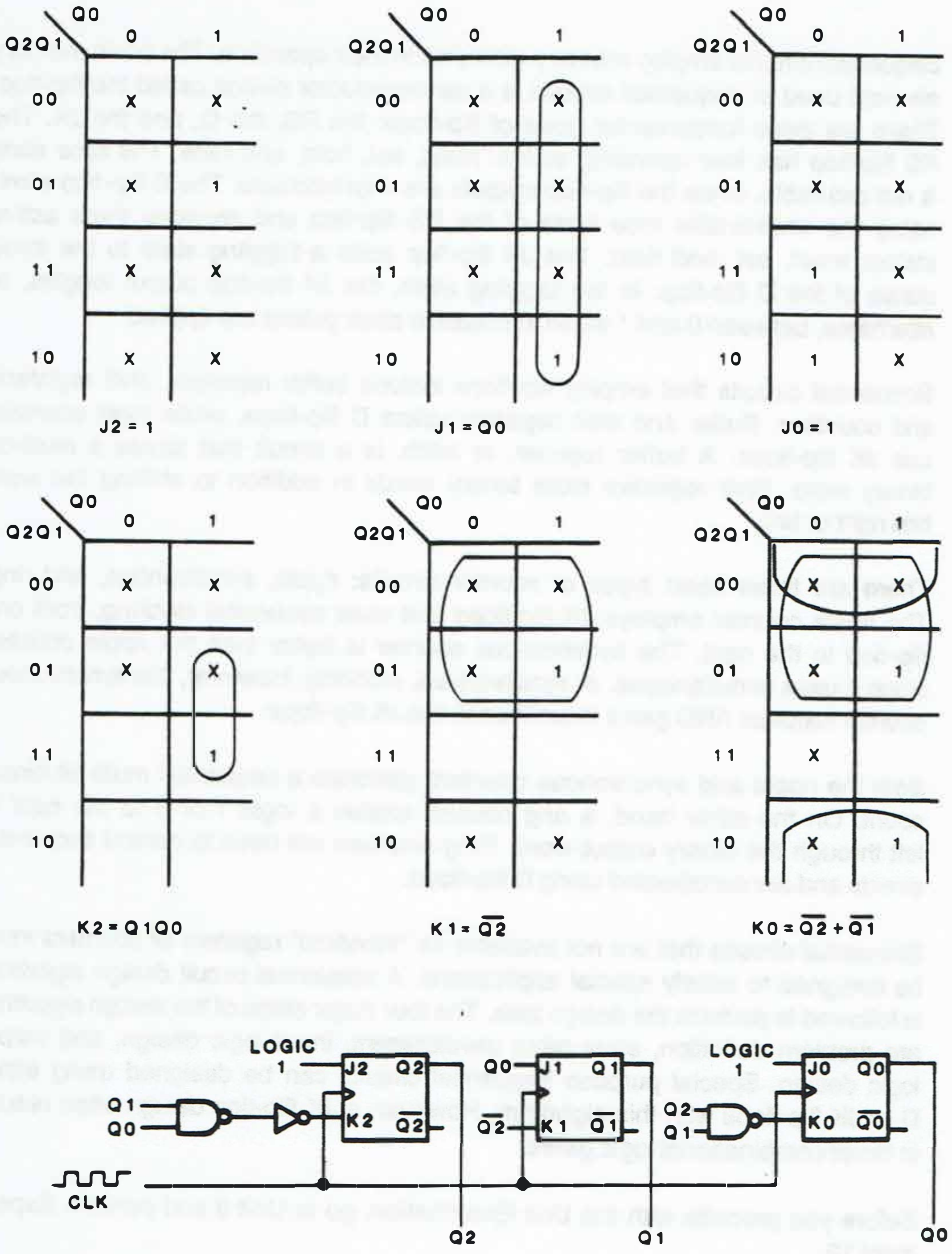


Figure 4-65
The JK flip-flop design for a 3-bit XS3 counter (Question 33).

SUMMARY

Sequential circuits employ memory elements in their operation. The basic memory element used in sequential circuits is a semiconductor device called the flip-flop. There are three fundamental types of flip-flops: the RS, the D, and the JK. The RS flip-flop has four operating states: reset, set, hold, and race. The race state is not desirable, since the flip-flop outputs are unpredictable. The D flip-flop eliminates the undesirable race state of the RS flip-flop and provides three active states: reset, set, and hold. The JK flip-flop adds a toggling state to the three states of the D flip-flop. In the toggling state, the JK flip-flop output toggles, or alternates, between 0 and 1 when successive clock pulses are applied.

Sequential circuits that employ flip-flops include buffer registers, shift registers, and counters. Buffer and shift registers utilize D flip-flops, while most counters use JK flip-flops. A buffer register, or latch, is a circuit that stores a multi-bit binary word. Shift registers store binary words in addition to shifting the word bits right or left.

There are three basic types of counter circuits: ripple, synchronous, and ring. The ripple counter employs JK flip-flops and uses sequential clocking, from one flip-flop to the next. The synchronous counter is faster than the ripple counter, since it uses simultaneous, or synchronous, clocking. However, the synchronous counter requires AND gates in addition to the JK flip-flops.

Both the ripple and synchronous counters generate a sequential multi-bit binary count. On the other hand, a ring counter rotates a logic 1 or 0 to the right or left through the binary output word. Ring counters are used to control sequential events and are constructed using D flip-flops.

Sequential circuits that are not available as "standard" registers or counters must be designed to satisfy special applications. A sequential circuit design algorithm is followed to perform the design task. The four major steps of the design algorithm are problem definition, state table development, input logic design, and output logic design. Special purpose sequential circuits can be designed using either D or JK flip-flops with this algorithm. However, a JK flip-flop design often results in fewer combinational logic gates.

Before you procede with the Unit Examination, go to Unit 8 and perform Experiment 13.

UNIT EXAMINATION

1. In the HOLD state, the outputs of a flip-flop:
   - A. Are unpredictable.
   - B. Are toggling.
   - C. Remain unchanged from the previous state.
   - D. Are all high.

2. If a flip-flop is racing, its outputs:
   - A. Are unpredictable.
   - B. Are toggling.
   - C. Remain unchanged from the previous state.
   - D. Are all high.

3. The symbol in Figure 4-66 denotes:
   - A. Positive edge triggering.
   - B. Negative edge triggering.
   - C. Positive level triggering.
   - D. Negative level triggering.

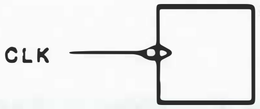


Figure 4-66

Symbol for Unit Exam Question 3.

4. The largest number that can be counted in a binary ripple counter is 2^n, where n is the number of flip-flop stages.
   - A. True
   - B. False

5. To set a JK flip-flop, its inputs must be:
   - A. J = K = 0.
   - B. J = K = 1.
   - C. J = 0, K = 1.
   - D. J = 1, K = 0.

6. Look at the circuit in Figure 4-67, and assume both flip-flops are initially reset. After two active clock pulses:
   - A. L0 = L1 = 0.
   - B. L0 = L1 = 1.
   - C. L0 = 0, L1 = 1.
   - D. L0 = 1, L1 = 0.

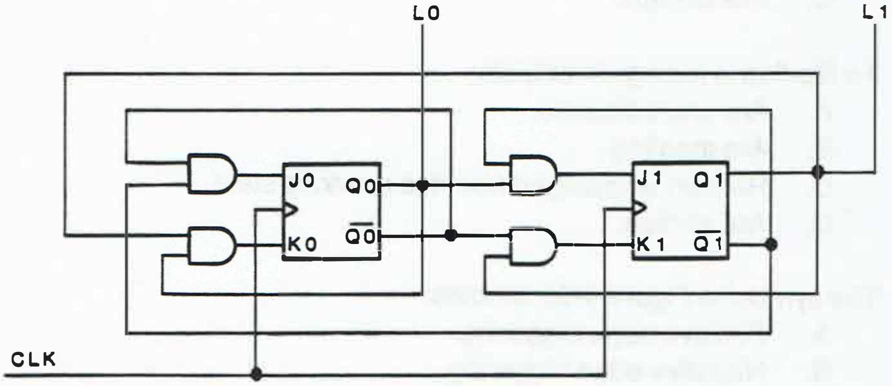


Figure 4-67

A flip-flop circuit for Unit Exam questions 6 and 7.

7. After four active clock pulses are applied to the circuit in Figure 4-67:
   - A. L0 = L1 = 0.
   - B. L0 = L1 = 1.
   - C. L0 = 0, L1 = 1.
   - D. L0 = 1, L1 = 0.

8. A sequential circuit used for converting between parallel and serial data formats is the:
   - A. Controlled buffer register.
   - B. Shift-left register.
   - C. Shift-right register.
   - D. Counter.

9. Seven flip-flops are connected to form a ripple counter. Assume the clock signal applied to the circuit is 512 kHz. If each flip-flop has a propagation delay of 20 nS, the frequency at the output of the 7th flip-flop is:
   - A. 73.14 kHz.
   - B. 256 kHz.
   - C. 4 kHz.
   - D. 2 kHz.

10. The period of the output signal from the second flip-flop in question 9 is:
    - A. 7.8 MHz.
    - B. 15.6 mS.
    - C. 7.8 mS.
    - D. None of these.

11. Assume 011 1111 is stored in the counter in question 9. How long after the next active clock edge arrives does it take to generate a count of 100 0000?
    - A. 20 nS.
    - B. 140 µS.
    - C. 140 nS.
    - D. 120 nS.

12. Look at the circuit in Figure 4-68. This circuit is a:
    - A. Ring counter.
    - B. Buffer register.
    - C. Shift register.
    - D. Ripple counter.

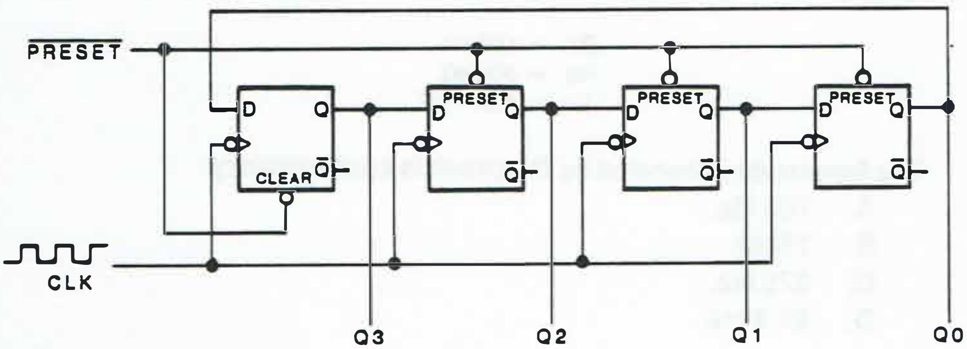


Figure 4-68

A flip-flop circuit for Unit Exam questions 12 and 13.

13. Assume the circuit in Figure 4-68 is initially $\overline{\text{PRESET}}$. After three active clock edges, the value of Q is:
    - A. 0111.
    - B. 1110.
    - C. 1101.
    - D. 0010.

14. Suppose a 6-bit presettable counter is preset to the value of 32. The modulus of this counter is:
    - A. 32.
    - B. 64.
    - C. 33.
    - D. 31.

15. Another name for a BCD counter is a:
    - A. Hex counter.
    - B. Divide-by-10 counter.
    - C. Divide-by-2 counter.
    - D. None of these.

16. A 7492 is a:
    - A. Hex counter.
    - B. Decade counter.
    - C. Divide-by-16 counter.
    - D. Divide-by-12 counter.

17. Suppose the 555 timer circuit in Figure 4-18 has the following component values:

    R1 = 100 kΩ
    R2 = 900 kΩ
    C = 0.001 μF

    The frequency generated by the circuit is approximately:
    - A. 750 Hz.
    - B. 75 Hz.
    - C. 375 Hz.
    - D. 37.5 Hz.

18. The duty cycle generated by the circuit in Figure 4-18 using the component values in question 17 is approximately:
    - A. 53%.
    - B. 50%.
    - C. 47%.
    - D. None of these.

19. Suppose you must design a 2-bit Gray Code counter using D flip-flops whose sequential outputs are 00, 01, 11, and 10. The equations for the D flip-flop inputs are:
    - A. $D1 = Q0, D0 = \overline{Q1}$.
    - B. $D1 = \overline{Q0}, D0 = Q1$.
    - C. $D1 = 0, D0 = 1$.
    - D. $D1 = Q0 + \overline{Q1}, D0 = \overline{Q0} + Q1$.

20. Using a JK design for question 19 requires J and K input equations of:
    - A. $J1 = Q0, K1 = \overline{Q0}, J0 = \overline{Q1}, K0 = \overline{Q1}$.
    - B. $J1 = \overline{Q1}, K1 = Q1, J0 = Q0, K0 = \overline{Q0}$.
    - C. $J1 = Q0, K1 = \overline{Q0}, J0 = \overline{Q1}, K0 = Q1$.
    - D. $J1 = Q0, K1 = \overline{Q0}, J0 = Q1, K0 = \overline{Q1}$.

EXAMINATION ANSWERS

1. C — In the HOLD state, the outputs of a flip-flop remain unchanged from the previous state.

2. A — If a flip-flop is racing, its outputs are unpredictable.

3. B — The symbol in Figure 4-66 denotes negative edge triggering.

4. B — The largest number that can be counted in a binary ripple counter is $2^n - 1$, not 2^n.

5. D — To set a JK flip-flop, its inputs must be $J = 1, K = 0$.

6. B — After two active clock pulses, the outputs of the circuit in Figure 4-67 are $L0 = L1 = 1$.

7. A — After four active clock pulses, the outputs of the circuit in Figure 4-67 are $L0 = L1 = 0$.

8. C — A sequential circuit used for converting between parallel and serial data formats is the shift-right register.

9. C — If seven flip-flops are connected to form a ripple counter, the frequency at the output of the 7th flip-flop is 4 kHz.

10. C — The period of the output signal from the second flip-flop in question 9 is 7.8 mS.

11. C — Assume 011 1111 is stored in the counter in question 9. It takes 140 nS after the next active clock edge arrives to generate a count of 100 0000.

12. A — The circuit in Figure 6-68 is a ring counter.

13. B — Assuming the circuit in Figure 6-68 is initially $\overline{\text{PRESET}}$, after three active clock edges the value of Q is 1110.

14. C — If a 6-bit presettable counter is preset to a value of 32, its modulus is 33.

15. B — Another name for a BCD counter is divide-by-10 counter.

16. D — A 7492 is a divide-by-12 counter.

17. C — If the 555 timer circuit in Figure 4-18 has the following component values:

R1 = 100 kΩ
R2 = 900 kΩ
C = 0.001 μF

The frequency generated by the circuit is approximately 375 Hz.

18. B — The duty cycle generated by the circuit in Figure 4-18 is always 50%.

19. A — The equations for the D flip-flop inputs of a 2-bit Gray code counter are $D1 = Q0$, $D0 = \overline{Q1}$.

20. C — Using a JK design for question 19 requires J and K input equations of $J1 = Q0$, $K1 = \overline{Q0}$, $J0 = \overline{Q1}$, $K0 = Q1$.

UNIT 5

ADDING I/O TO THE DESIGN

CONTENTS

Introduction . . . . Page 5-3

Unit Objectives . . . . Page 5-4

Unit Activity Guide . . . . Page 5-6

Input Devices . . . . Page 5-7

Display Devices . . . . Page 5-24

Control Devices . . . . Page 5-61

Summary . . . . Page 5-88

Unit Examination . . . . Page 5-89

Examination Answers . . . . Page 5-95

INTRODUCTION

The combinational and sequential logic circuits that you have been designing up to this point are primarily used to process binary data, once it is available to the circuit. You have not had to consider how to input the data or how to output the results of the logic circuit processing to the outside world. In other words, up to this point you have completely ignored any input/output, or I/O, considerations for the circuit, except to define the I/O logic. It is now time for you to learn how to add I/O to your digital circuit design.

The first section of this unit covers the input side of the circuit. Input devices can be broken down into two general categories: those used by humans to enter binary data into a digital circuit, and those devices that automatically sense and transmit external conditions to the digital circuit. A computer terminal keyboard is an example of a human input device, while an electronic thermostat is an example of an automatic sensing device. In this text, you will learn about switches and keyboards, the human side of input devices.

The last two sections of this unit are devoted to output devices. Again, output devices can be broken down into two categories: those that are used to display information to a human operator, and those that are used by the digital circuit to automatically control some external operation. You will first learn all about digital display devices and how to design display circuits. Then, you will be acquainted with several output devices used for industrial and commercial applications to control external operations. Now, let's get to work!

UNIT OBJECTIVES

When you complete this unit, you should be able to:

1. Design switching circuits that can be used to input binary data into a digital circuit.
2. List several different ways that debouncing can be provided for mechanical switches.
3. Explain how digital keyboards are constructed.
4. Design keyboard decoding circuits.
5. Describe the operation of the following display devices:
    A. LED displays.
    B. Incandescent displays.
    C. Gas discharge displays.
    D. Vacuum fluorescent displays.
    E. LCD displays.
    F. Dot matrix displays.
6. List several TTL and CMOS decoder/driver devices that are used to drive the above displays.
7. Design control circuits for the above displays.
8. Explain how to eliminate leading zeros in a multidigit display.

9. State why driver transistors are often required in display circuits.

10. Discuss the operation of a multiplexed display.

11. Describe the operation of the following control devices:

    A. Open collector devices.
    B. Peripheral power drivers.
    C. Transistor arrays.
    D. Solid-state relays.
    E. Reed and electro-mechanical relays.
    F. Optoisolators.

12. Design digital control circuits using the above devices.

13. Protect a digital circuit and its human operator from high power load circuits using optical isolation.

14. Design circuits that will convert between TTL, RS232C, and 20 mA current loop serial data transmission standards.

UNIT ACTIVITY GUIDE

☐ Read section on “Input Devices.”

☐ Answer Self-Test Review Questions 1-10.

☐ Read section on “Display Devices.”

☐ Answer Self-Test Review Questions 11-20.

☐ Perform Experiment 14 in Unit 8.

☐ Read section on “Control Devices.”

☐ Answer Self-Test Review Questions 21-29.

☐ Study Summary.

☐ Complete Unit Examination.

☐ Check Examination Answers.

INPUT DEVICES

The simplest of all digital input devices is the mechanical switch. In fact, many complex input devices, such as keyboards, are simply a collection of switches. In this section, you will begin by reviewing how simple mechanical switches can be used to apply binary logic to digital circuits. This discussion will then be extended to include various multiple switch keyboard arrangements.

Mechanical Switches

Digital, or binary, switches are two-state mechanical devices as shown in Figure 5-1. Notice that each switch can only assume one of two states, open or closed, A or B. The switch symbol in Figure 5-1a is a pushbutton switch, like those used in keyboards for entering data. The switch in Figure 5-1b is called a single-pole/single-throw, or SPST, switch, while Figure 5-1c shows a single-pole/double-throw, or SPDT, switch.

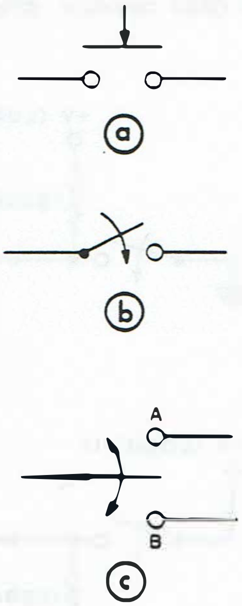


Figure 5-1

Mechanical switches used for binary input: pushbutton (a), single-pole/single-throw (b), single-pole/double-throw (c).

CIRCUIT APPLICATIONS

Pushbutton and SPST switches are used to apply logic to a digital circuit via a pull-up or pull-down resistor circuit as shown in Figure 5-2. When the switch is open in a pull-up resistor circuit the circuit generates a logic 1, since the path to ground is broken and the resistor doesn't drop any voltage. Consequently, all the supply voltage is seen at the bottom of the pull-up resistor. When the switch is closed, the switch makes a path to ground and the pull-up resistor drops all the supply voltage. Thus, a ground, or zero, potential is seen at the bottom of the resistor. Hence, the circuit generates a logic 0.

The action of a pull-down circuit is just the opposite as that of a pull-up circuit. When the switch is open, the circuit generates a logic 0, since a ground potential is seen at the top of the pull-down resistor. When the switch is closed the circuit generates a logic 1, since the supply potential is seen at the top of the resistor.

Clearly, the supply potential, +V, in Figure 5-2 must be the level required by the digital circuit for a logic 1. For TTL logic, it must be +5 volts, while CMOS logic requires between +3 and +18 volts. The value of the pull-up or pull-down resistor is not critical. I use a 1000 ohm resistor, since they are low cost and readily available.

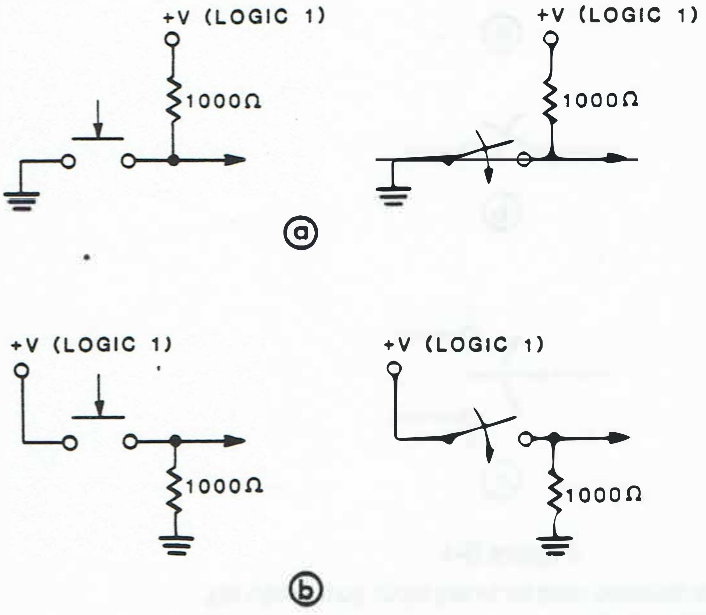


Figure 5-2
Pull-up (a) and pull-down (b) resistor circuits are used with pushbutton and SPST switches.

A single-pole/double-throw switch is used to apply logic to a digital circuit as shown in Figure 5-3. Here, two pull-up or pull-down resistors are used, one for each switch terminal. The operating principle of the resistor circuit is the same as described earlier. The pull-up circuit in Figure 5-3a applies a logic 0 to one of the two terminals, A or B, depending on the respective switch position. The opposite terminal is "pulled-up" to a logic 1 state. The pull-down circuit in Figure 5-3b applies a logic 1 to the respective switch position terminal, while the opposite terminal is "pulled-down" to a logic 0 state.

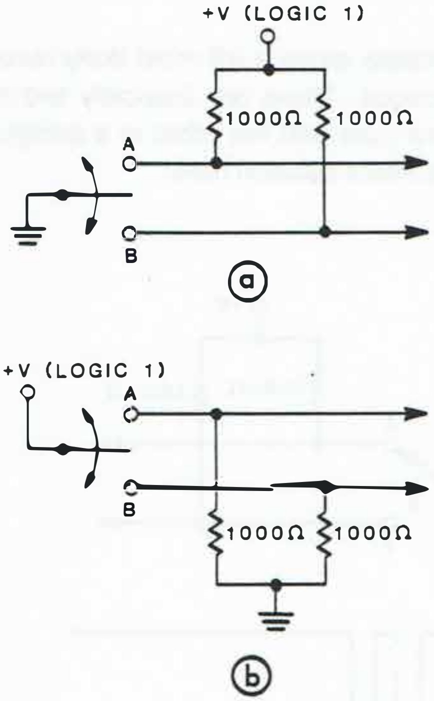


Figure 5-3

With SPDT switches, a logic 0 (a) or a logic 1 (b) is applied to one of two positions, A or B.

MECHANICAL SWITCH DEBOUNCING

The problem of mechanical switch bouncing is illustrated by Figure 5-4. Look at what happens to the logic signal when the switch is moved from position A to position B. The **bounce regions** result from vibration of the mechanical switch contacts. Observe that, as the contacts vibrate, a series of 1's and 0's are generated on the signal line. This is due to the make/break action of the contacts. The bouncing period is typically 10 mS but can be as high as 100 mS, depending on the switch. Digital switch manufacturers will usually specify a maximum bounce period for their products.

Bouncing is highly undesirable, since it will most likely result in the false triggering of a subsequent digital circuit. There are basically two cures for the bouncing ailment: one is a hardware cure and the other is a computer controlled software cure. Let's discuss the hardware solution here.

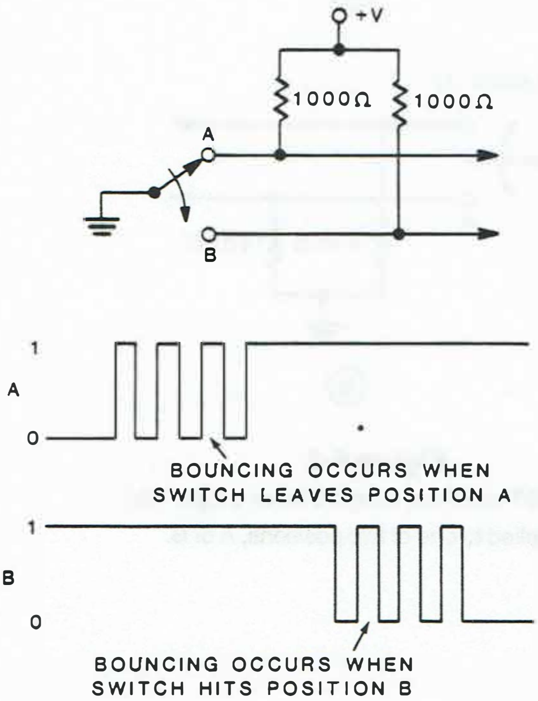
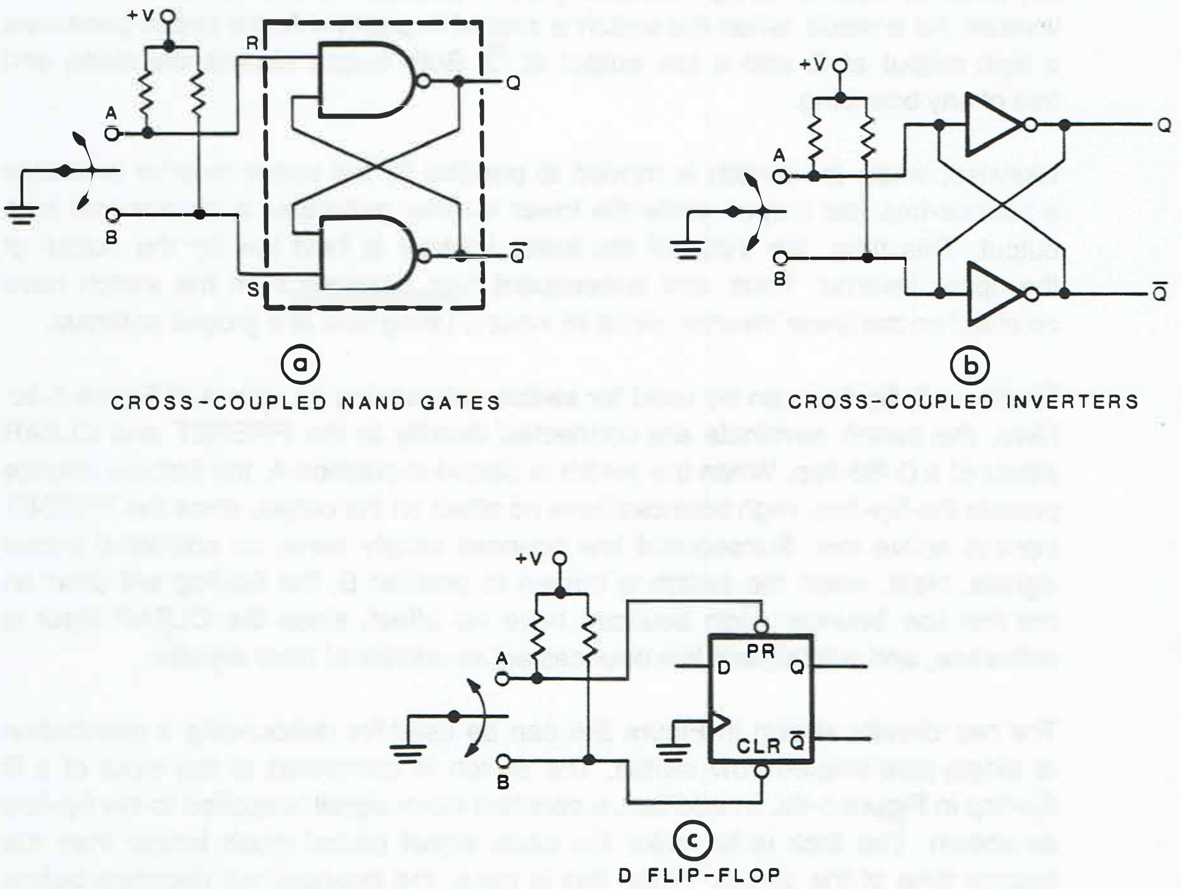


Figure 5-4

Mechanical vibration of switch contacts causes the switch output to bounce as shown.

The elimination of switch bouncing is called **switch debouncing**. You can use any of the circuits shown in Figure 5-5 to debounce a single-pole/double-throw switch. You should recognize the cross coupled NAND gate circuit in Figure 5-5a as an RS flip-flop. When the switch is placed in position A, the S input goes high and the R input begins to bounce between low and high. The first time the contacts bounce low the circuit will set, since RS = 01. When the contacts bounce high the circuit output will not change from the set state, since RS = 11 puts the flip-flop in its hold state. Thus, subsequent bounces have no effect on the circuit output.



CROSS-COUPLED NAND GATES

CROSS-COUPLED INVERTERS

D FLIP-FLOP

Figure 5-5

Three types of switch debouncing circuits for SPDT switches.

When the switch is moved to position B, the R input goes high and the S input begins to bounce between low and high. The first time the circuit bounces low, the circuit will reset, since RS = 10. The circuit remains reset, since any subsequent high bounces place the flip-flop in its hold state.

A simple switch debouncer can be constructed using two inverters as shown in Figure 5-5b. When the switch is thrown to position A, the upper inverter input bounces between low and high, while a high level is applied to the lower inverter. As a result, the lower inverter generates a constant low output. This low output is applied directly to the input of the upper inverter and holds the input low. Any subsequent high bounces from the switch have no effect on the upper inverter, since its input is being held at a ground potential by the output of the lower inverter. As a result, when the switch is placed in position A, the circuit generates a high output at Q and a low output at $\overline{Q}$. Both output signals are clean and free of any bouncing.

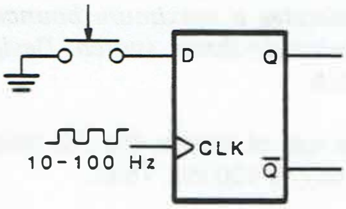
Likewise, when the switch is moved to position B, the upper inverter produces a bounce-free low output, while the lower inverter generates a bounce-free high output. This time, the input of the lower inverter is held low by the output of the upper inverter. Thus, any subsequent high bounces from the switch have no effect on the lower inverter, since its input is being held at a ground potential.

Finally, a D flip-flop can be used for switch debouncing as shown in Figure 5-5c. Here, the switch terminals are connected directly to the PRESET and CLEAR inputs of a D flip-flop. When the switch is placed in position A, the first low bounce presets the flip-flop. High bounces have no effect on the output, since the PRESET input is active low. Subsequent low bounces simply serve as additional preset signals. Next, when the switch is thrown to position B, the flip-flop will clear on the first low bounce. High bounces have no effect, since the CLEAR input is active low, and subsequent low bounces act as additional clear signals.

The two circuits shown in Figure 5-6 can be used for debouncing a pushbutton or single-pole/single-throw switch. The switch is connected to the input of a D flip-flop in Figure 5-6a. In addition, a constant clock signal is applied to the flip-flop as shown. The trick is to make the clock signal period much longer than the bounce time of the switch. When this is done, the bounces will dissipate before an active clock edge hits. A clean bounceless signal will then be clocked to the flip-flop output.

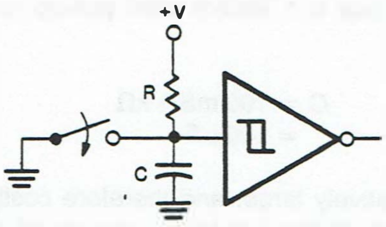
The 7414 in Figure 5-6b is called a **Schmitt-trigger** inverter. Schmitt-triggered gates are designated with a hysteresis symbol within the gate. The operation of a Schmitt-triggered gate is such that it will produce a rectangular output, regardless of variations on the input signal. Thus, Schmitt-triggered gates are used to "clean-up" noisy signals. The 7414 is a hex Schmitt-trigger inverter, meaning that there are six inverters in one IC package. There are only two more Schmitt trigger ICs in the 7400 series: the 7413 dual 4-bit NAND, and the 74132 quad 2-bit NAND.

Now, the switch debouncing circuit in Figure 5-6b shows a capacitor connected to the input of a Schmitt-trigger inverter. The product of the pull-up resistor, R, and the capacitor, C, is called the **RC time constant**. The RC time constant determines how long it takes the capacitor (C) to charge. Larger time constants result in a longer charge time.



ⓐ

D FLIP-FLOP



ⓑ

SCHMITT-TRIGGER

Figure 5-6

Two ways to debounce a pushbutton or SPST switch.

The idea is to make the RC time constant longer than the switch bounce time. To see how it works, assume the switch is open. This allows the capacitor to charge to a logic 1 level from the supply potential. As a result, a logic 1 is seen at the input of the inverter. Since the inverter sees a high input, it generates a low output. Next, when the switch is closed, the first low bounce connects the top plate of the capacitor to ground and discharges it very rapidly through the switch. Thus, a low level is seen at the inverter input, causing a high level output. Subsequent high bounces do not have enough time to charge the capacitor back to a logic 1 level, because of the long RC time constant. Consequently, the inverter output remains high and bounce-free.

Example 5-1

A good rule of thumb is to design a Schmitt-trigger debouncing circuit such that the input RC time constant is at least ten times the switch bounce period. Suppose a manufacturer's specification indicates a maximum bounce period of 10 mS for a single-pole/single-throw switch. Design a circuit to debounce this switch.

Using the above rule of thumb, the RC time constant must be at least 10 × 10 mS, or 100 mS. Thus,

$$RC = 100 \text{ mS}$$

or,

$$C = 100 \text{ mS/R}$$

Now, if you use a 1 kilohm ohm pull-up resistor, the value of C must be:

$$\begin{aligned} C &= 100 \text{ mS}/1 \text{ k}\Omega \\ &= 100 \,\mu\text{F} \end{aligned}$$

This is a relatively large, and therefore costly, capacitor. To reduce the size of the capacitor, you could use a 10 kilohm or 100 kilohm pull-up resistor. These values would require a 10 μF or 1 μF capacitor, respectively.

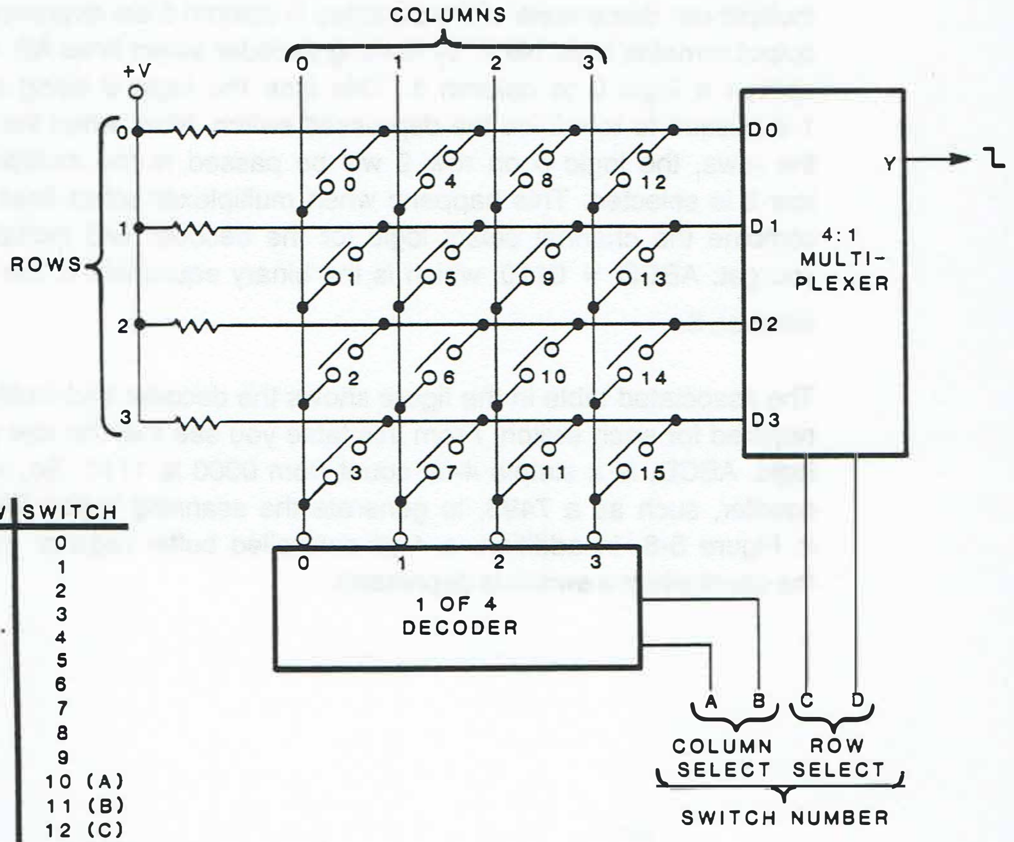
Note that as long as you are dealing in kilohm resistor values and microfarad capacitor values, the time constant will be in milliseconds.

Before leaving switch debouncing, I should mention that there are so called **bounceless switches** available. As the name implies, such switches do not require debouncing because they do not use mechanical contacts. Instead, they operate on either a magnetic or capacitive principle. Of course, the trade-off is a higher price.

Keyboards

A keyboard is simply a matrix of switches. There are decimal keyboards containing ten switches, hex keyboards with sixteen switches, and ASCII keyboards with 64 or more switches. Regardless of its size, the design and operation of all keyboards are basically the same. A typical hex keyboard switch matrix is shown in Figure 5-7. Notice that the switch matrix includes 4 rows and 4 columns. Sixteen pushbutton switches provide the connections between the rows and columns.

The task before us is to design a circuit that will detect a switch closure and determine which particular switch is closed. Such a circuit is called a **keyboard decoding** circuit. As you can see from Figure 5-7, a keyboard decoding circuit can be designed using a decoder and a multiplexer.



ABCD	COL.	ROW	SWITCH
0000	0	0	0
0001	0	1	1
0010	0	2	2
0011	0	3	3
0100	1	0	4
0101	1	1	5
0110	1	2	6
0111	1	3	7
1000	2	0	8
1001	2	1	9
1010	2	2	10 (A)
1011	2	3	11 (B)
1100	3	0	12 (C)
1101	3	1	13 (D)
1110	3	2	14 (E)
1111	3	3	15 (F)

Figure 5-7

A hex keyboard and associated decoder circuit.

First, notice that the pushbuttons are connected to the supply via pull-up resistors. Consequently, logic 1's appear at the multiplexer inputs as long as no switches are depressed. Now, the idea is to successively apply a logic 0 to each column using the decoder. After a logic 0 is applied to a given column, the rows are scanned by the multiplexer. This is done by selecting each multiplexer input, in their respective order, using the multiplexer channel select lines. If a switch in a given column is depressed, the logic 0 from the decoder will be passed, via the switch, to one of the multiplexer input rows. When that particular input row is selected, the logic zero will appear at the multiplexer output. The combined binary value present at that time on the decoder and multiplexer select lines will indicate which particular switch is depressed.

For example, suppose that switch 6 is depressed. Notice that this switch connects column 1 to row 2. To detect the switch closure, a logic 0 is first applied to column 0 by decoder select lines A and B. Then, the rows are scanned by the multiplexer. Since none of the switches in column 0 are depressed, the multiplexer output remains high. Next, by making decoder select lines AB = 01, the decoder applies a logic 0 to column 1. This time the logic 0 being applied to column 1 is passed to row 2 via the depressed switch. Now, when the multiplexer scans the rows, the logic 0 on row 2 will be passed to the multiplexer output when row 2 is selected. This happens when multiplexer select lines CD = 10. If you combine the channel select logic for the decoder and multiplexer at this time you get: ABCD = 0110, which is the binary equivalent of the depressed switch number, 6.

The associated table in the figure shows the decoder and multiplexer select logic required for each switch. From this table you see that the row and column select logic, ABCD, is a simple 4-bit count from 0000 to 1111. So, why not use a hex counter, such as a 7493, to generate the scanning logic? This has been done in Figure 5-8. In addition, a 4-bit controlled buffer register was added to latch the count when a switch is depressed.

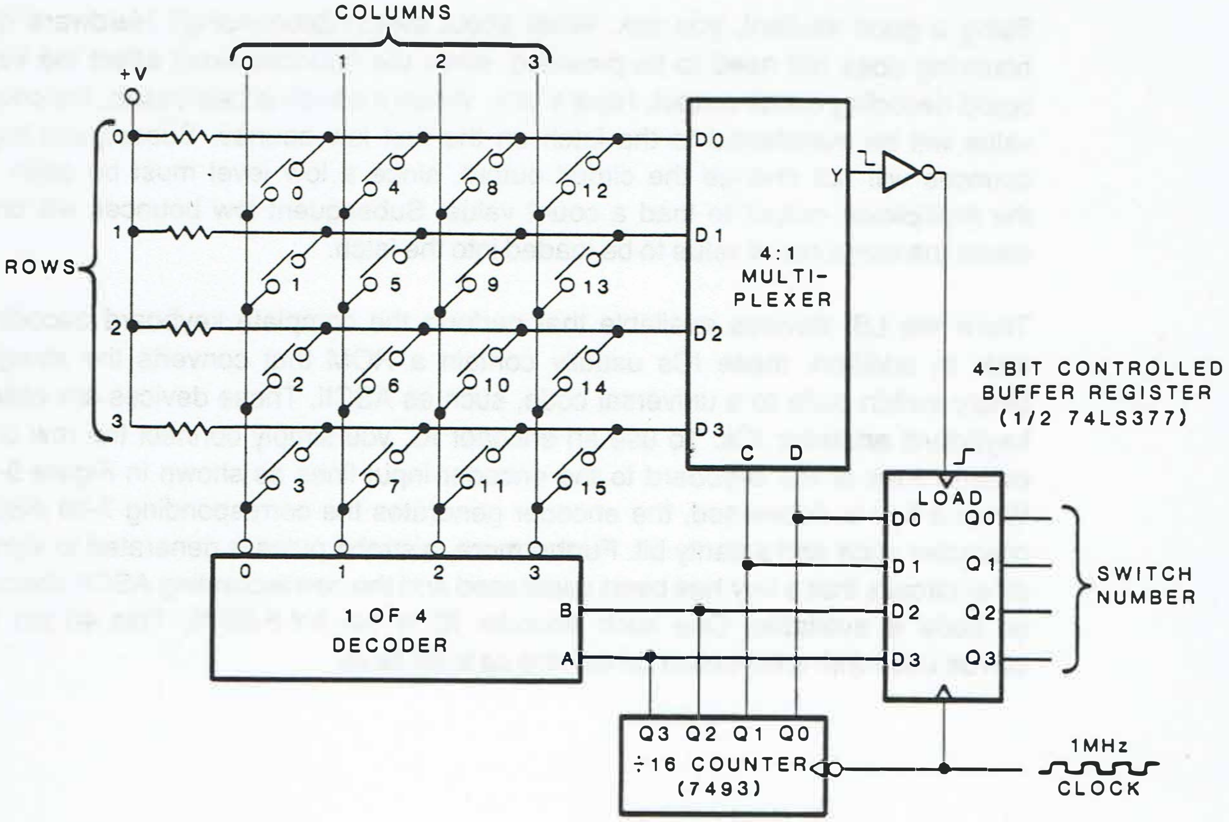


Figure 5-8

A complete keyboard decoding circuit.

Here's how it works. The counter runs freely, generating the row and column select logic. When a key is depressed, the multiplexer output activates the LOAD line of the 4-bit controlled buffer register. The count at this time is then loaded and stored by the buffer register until another key is depressed. Of course, timing is critical. The clock frequency must be high enough to catch any momentary switch closures. A 1 MHz clock signal should do the job. In addition, notice that the counter is negatively edge triggered and the buffer register is positively edge triggered. This is done so that the count won't increment before it can be stored by the buffer register. A given count value will be latched on the next positive edge, before a negative edge can increment the counter.

Being a good student, you ask: What about switch debouncing? Hardware debouncing does not need to be provided, since the bounces won't affect the keyboard decoding circuit output. Here's why. When a switch is depressed, the count value will be transferred to the latch on the first low bounce. Subsequent high bounces will not change the circuit output, since a low level must be seen at the multiplexer output to load a count value. Subsequent low bounces will only cause the same count value to be loaded into the latch.

There are LSI devices available that perform the complete keyboard decoding task. In addition, these ICs usually contain a ROM that converts the straight binary switch code to a universal code, such as ASCII. These devices are called **keyboard encoder** ICs. To use an encoder IC, you simply connect the row and column lines of the keyboard to the encoder input lines as shown in Figure 5-9. When a key is depressed, the encoder generates the corresponding 7-bit ASCII character code and a parity bit. Furthermore, a strobe pulse is generated to signal other circuits that a key has been depressed and the corresponding ASCII character code is available. One such encoder IC is the AY-5-2376. This 40 pin IC can be used with a keyboard containing up to 88 keys.

Before leaving keyboards, you should be aware that there are basically two types of keyboards commercially available: **unencoded** and **fully encoded** keyboards. In their simplest form, unencoded keyboards are nothing more than a switch matrix. You must supply the keyboard decoding logic. On the other hand, fully encoded keyboards contain all the required decoding logic. As a result, this type of keyboard will generate a unique code for each key. Most fully encoded keyboards use a typewriter key arrangement and generate ASCII code to be compatible with different digital systems.

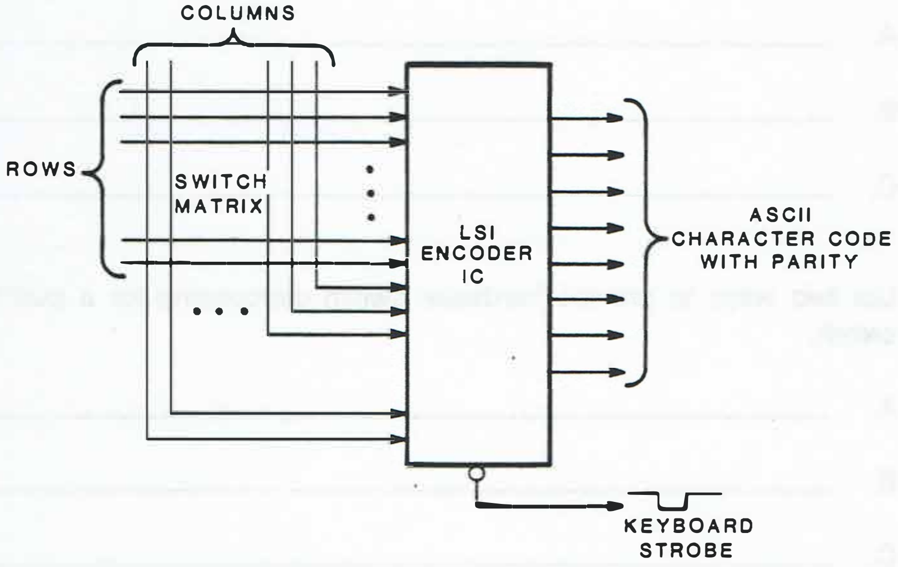


Figure 5-9
LSI encoder ICs are available to perform the complete keyboard decoding task.

Self-Test Review

1. The three general types of mechanical switches used to apply logic to a digital circuit are the ________, ________, and ________ switches.

2. A closed pushbutton switch in a pull-up resistor circuit generates a logic ________ level.

3. List three ways to provide hardware switch debouncing for a SPDT switch.

   A. ____

   B. ____

   C. ____

4. List two ways to provide hardware switch debouncing for a pushbutton switch.

   A. ____

   B. ____

   C. ____

5. Design a circuit to debounce a SPDT switch using cross-coupled NOR gates.

6. A manufacturer's data sheet specifies a maximum bounce period of 5 mS for a pushbutton switch. What size resistor must be used with a 0.01 μF capacitor to debounce this switch using a Schmitt-trigger circuit? ____________

7. The two MSI devices required to decode a keyboard are a ____________ and ________________.

8. Row scanning of a switch matrix is accomplished using a ____________ .

9. Suppose the 7493 counter in Figure 5-8 has a count of 1101 when LOAD goes high.

   A. Has a switch been depressed? ________________

   B. If so, which one? ________________

10. Explain the difference between an unencoded keyboard and a fully encoded keyboard. ____
____
____

Answers

1. pushbutton, single-pole/single-throw (SPST), and single-pole/double throw (SPDT)

2. 0 (zero)

3A. Cross-coupled NAND gates.

3B. Cross-coupled inverters.

3C. D flip-flop.

4A. D flip-flop.

4B. Schmitt-trigger circuit.

5. Solution follows.

6. 5 MΩ

7. decoder, multiplexer

8. multiplexer

9A. Yes

9B. A binary count of 1101 is equivalent to a decimal count of 13. This means that switch 13 has been depressed.

10. An unencoded keyboard is nothing more than a switch matrix which must be decoded. Fully encoded keyboards contain all the required decoding logic.

Solution to Questions 5 and 6.

5. The difference between an RS flip-flop using NAND gates and an RS flip-flop using NOR gates is that the HOLD and RACE states are reversed. With a cross-coupled NOR gate circuit, the HOLD state occurs when RS = 00 and the RACE state occurs when RS = 11. As a result, the input logic to a cross-coupled NOR gate switch debouncing circuit must be reversed from that of a cross-coupled NAND gate circuit. To do this, a pull-down resistor circuit is used as shown in Figure 5-10, rather than a pull-up resistor circuit.

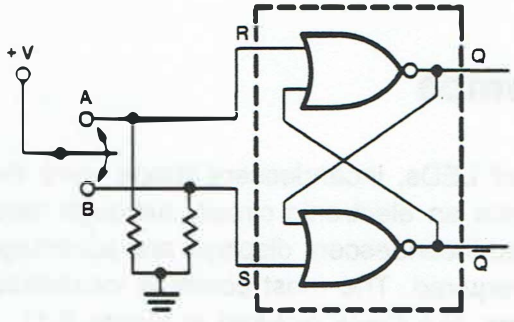


Figure 5-10
A cross-coupled NOR gate debouncing circuit requires that pull-down resistors be used on the circuit input (Question 5).

6. The RC time constant must be at least ten times the bounce period. Thus, the time constant is a minimum 10 × 5 mS, or 50 mS. If C = 0.01 µF, then:

$$R \times 0.01\ \mu F = 50\ mS$$

or,

$$\begin{aligned} R &= 50\ mS/0.01\ \mu F \\ &= 5000 \times 10^{3} \\ &= 5 \times 10^{6} \\ &= 5\ M\Omega \end{aligned}$$

DISPLAY DEVICES

In this section, you will learn about electronic devices that are used to display the output of a digital circuit. Display devices are used to provide information to the user of a digital circuit. A display device might indicate the status of a machine to an operator, or tell you how many gallons of gas you have pumped into your gas tank. You have already been acquainted with several display devices, such as single LEDs and 7-segment LEDs. Now it is time to take a closer look at the characteristics of these devices, and to add to your knowledge of displays by looking at some other common display devices like incandescent lamps, gas discharge, vacuum fluorescent and LCD displays.

Incandescent Lamps

Before the introduction of LEDs, incandescent lamps were the sole means of displaying information from an electronic circuit. Although most displays today are LED or LCD displays, incandescent displays are advantageous where high intensity illumination is required. The most common incandescent displays are single incandescent lamps, like those pictured in Figure 5-11. As you can see, these devices come in various shapes and sizes. Furthermore, they are available in many different colors, such as red, green, yellow, clear, and blue, just to mention a few.

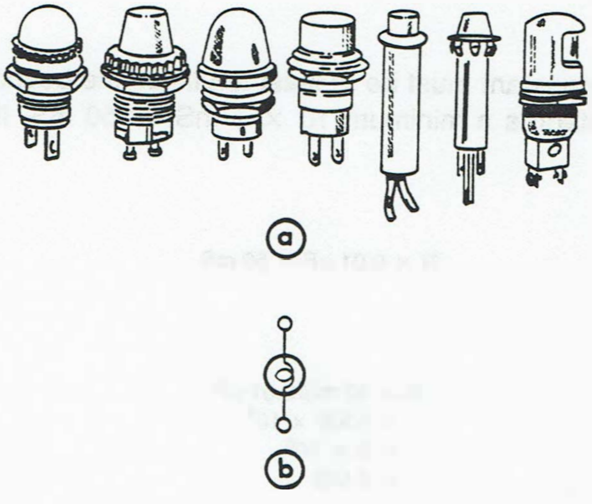


Figure 5-11

Various incandescent lamps (a) and associated symbol (b).

The problem in using incandescent lamps with digital circuits is that standard TTL and CMOS outputs do not have sufficient power to drive, and thus illuminate, the lamp. The solution to this problem is to use open collector gates at the circuit output to drive the lamps. Recall that an open collector output acts like a switch in a pull-up resistor circuit as shown again in Figure 5-12. When the switch is open, there is an infinite resistance path to ground and the circuit output is pulled-high to the +V potential. However, when the switch is closed, there is a low resistance path to ground through the open collector device. As a result, the circuit output is at ground potential, or 0 volts. In summary, you could say that the open collector device makes or breaks the circuit path to ground.

Notice that when the switch is closed, current is **sinked** to ground through the open collector device. The amount of current that can be sinked is limited by the current-carrying capabilities of the internal integrated transistors.

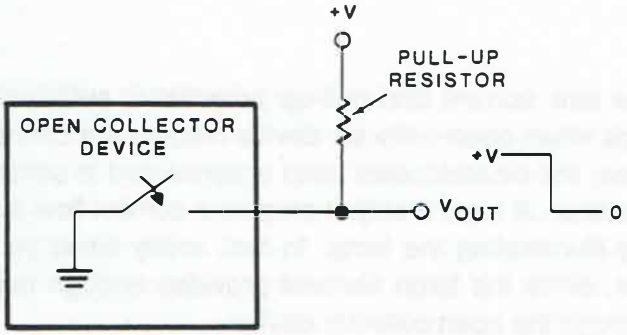


Figure 5-12

An open collector output acts like a switch in a pull-up resistor circuit.

Logic devices that provide open collector outputs are sometimes called open collector **drivers**. Open collector drivers are rated by their maximum current-sinking capabilities when the output is low, and the maximum pull-up potential for the high output condition. Several popular 7400-series open collector drivers are listed in Table 5-1, along with their maximum ratings.

Table 5-1: Several Popular 7400-Series Open Collector Devices

Open Collector Device	Maximum Sink Current	Maximum Pull-Up Potential
7405 Hex Inverter	12 mA	+ 5 V
7406 Hex Inverter	40 mA	+ 30 V
7407 Hex Buffer	40 mA	+ 30 V
7426 Quad 2-Bit NAND	16 mA	+ 15 V
7433 Quad 2-Bit NOR	48 mA	+ 5 V

In most cases, the sink current and pull-up potential is sufficient to drive small incandescent lamps when open collector device outputs are connected as shown in Figure 5-13. Here, the incandescent lamp is connected in series with the open collector pull-up resistor. A logic 0 output creates a current flow through the lamp to ground, thereby illuminating the lamp. In fact, many times you can eliminate the pull-up resistor, since the lamp filament provides enough resistance to limit the current flow through the open collector device.

There are three popular voltage ratings for small incandescent lamps: 6 volts, 12 volts, and 28 volts. Current ratings for these devices range from 20 to 60 milliamps, with the lower voltage ratings requiring a higher current rating to achieve the same relative brightness level. You must pick an incandescent lamp and open collector device such that their ratings are compatible to achieve optimum brightness, without damaging the lamp or open collector device.

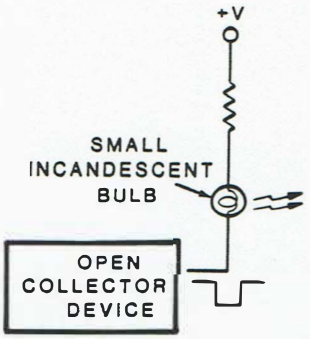


Figure 5-13

Open collector devices are used to drive incandescent lamps.

Example 5-2

Three different ratings for a given incandescent lamp are shown in a manufacturer's catalog as follows:

Type	*Voltage*	*Current*
1	*4-6 V*	*60 mA*
2	*12-14 V*	*40 mA*
3	*24-28 V*	*24 mA*

A. Which lamp type must be used with a 7405 Hex Inverter to achieve optimum brightness?

B. Which lamp type must be used with a 7406 Hex Inverter to achieve optimum brightness?

C. Which lamp type must be used with a 7407 Hex Buffer to achieve optimum brightness?

Matching the open collector device ratings in Table 5-1 to these incandescent lamp ratings you get:

A. None of the lamps listed will provide optimum brightness with a 7405 Hex Inverter, since the 7405 cannot sink the required amount of current.

B. The 7406 Hex Inverter could be used to drive the Types 2 and 3 lamps and achieve optimum brightness, since the ratings of these devices are compatible. It could also be used with the Type 1 lamp but the illumination would not be optimum, since the 7406 cannot sink the required amount of current. Of course, the pull-up potential must be compatible with the voltage rating of the lamp.

C. The 7407 Hex Buffer could be used with the Type 2 and Type 3 lamps. When using a Type 2 lamp the pull-up potential must be from 12 to 14 volts. When using a Type 3 lamp the supply potential must be from 24 to 28 volts and the current must be limited to 24 milliamps with a pull-up resistor. The Type 1 lamp could be used with a +5-volt pull-up potential, but illumination would not be optimum, since the 7407 can only sink 40 milliamps.

Discrete LEDs

Like incandescent lamps, discrete LEDs can be used to indicate status information. In addition, single LEDs can be arranged to generate various patterns, generate binary data, or display simple messages. Most LEDs do not provide the brightness levels of incandescent lamps, but they require less power to operate and are usually less expensive.

Discrete LEDs are small, usually less than 1/4 inch in diameter, and come in a variety of mounting configurations as shown in Figure 5-14. The most common colors are red, green, and yellow.

Unlike incandescent lamps, LEDs do not require open collector drivers to operate. A typical operating current is 20 mA, but they will also illuminate with less current. This is why a standard TTL output of 16 mA can be used to drive an LED directly. Of course, open collector devices can also be used to achieve greater brightness levels.

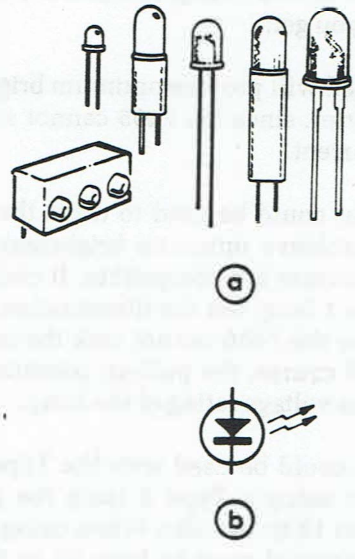


Figure 5-14
Various discrete LEDs (a) and symbol (b).

Two ways to connect an LED are shown in Figure 5-15. The LED in Figure 5-15a has its anode connected to the supply, thereby requiring a logic 0 to illuminate. A TTL, CMOS, or open collector TTL output can be used to drive the LED. The pull-up resistor is used to limit the current, thereby preventing damage to the LED and logic device. The resistor value is usually below 1000 ohms, with values of 150 and 333 ohms being typical. This value can be adjusted to vary the LED brightness level, but should always limit the current to the maximum level specified by the LED manufacturer.

In Figure 5-15b, the LED has its cathode connected to ground, requiring a logic 1 to illuminate. This circuit is limited to non-open collector TTL and CMOS outputs, since no pull-up potential is provided. In addition, many times no current limiting resistor is needed, since the internal resistance of the driving device is sufficient to limit the current through the LED. However, a current limiting resistor should be used when CMOS devices are operated above +5 volts.

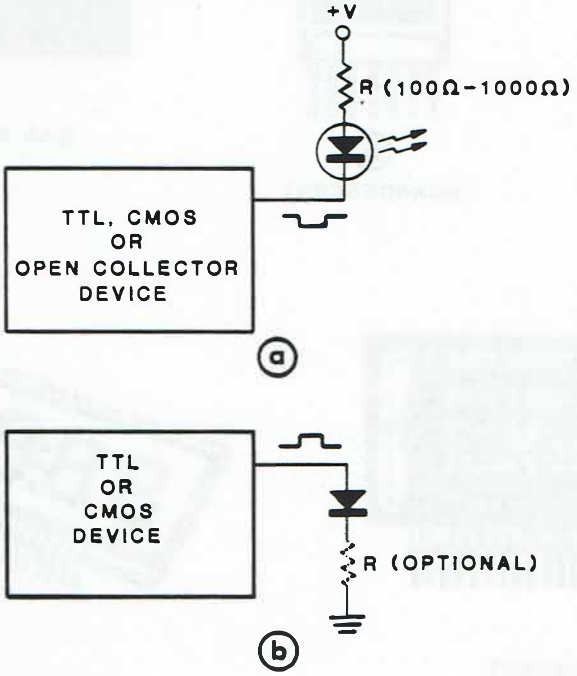
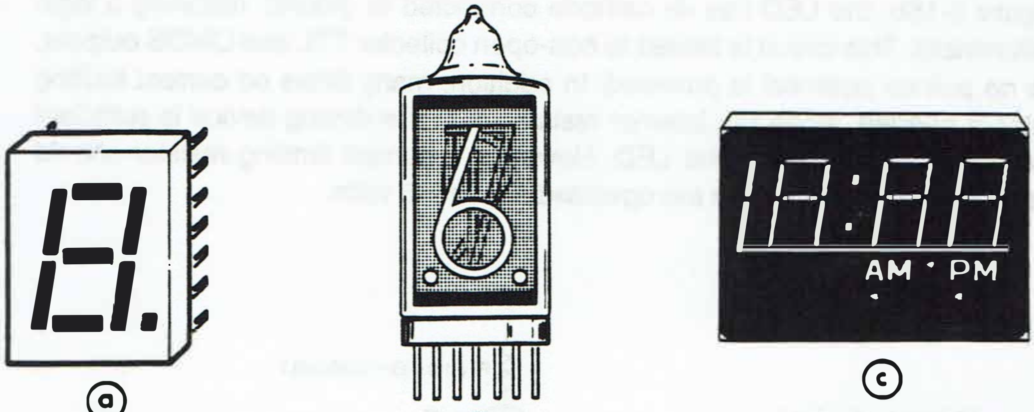


Figure 5-15

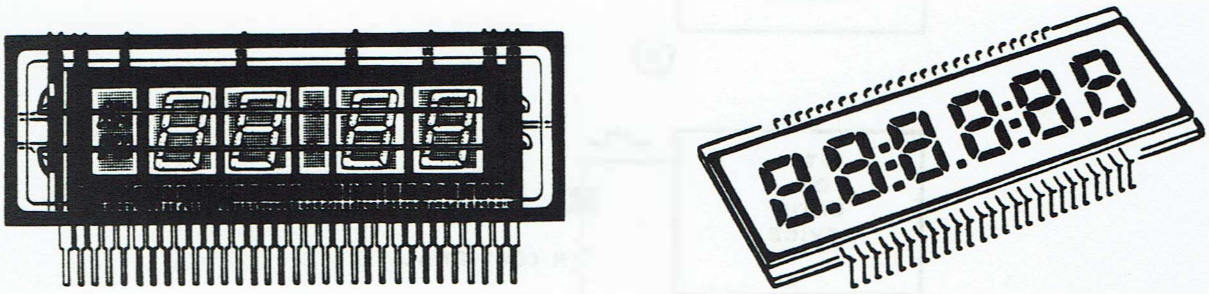
Discrete LEDs can be connected in two ways:
with anode to the supply (a), or cathode to ground (b).

Segmented Displays

As you are aware, segmented displays are widely used in electronic equipment such as digital clocks, calculators, and test instruments. Several types of segmented displays are pictured in Figure 5-16. These five popular display types are: **LED, Incandescent, gas discharge, vacuum fluorescent, and LCD**. You have probably seen all of these at one time or another. Let's take a brief look at each type.



ⓐ LED ⓑ INCANDESCENT ⓒ GAS DISCHARGE



ⓓ VACUUM FLUORESCENT ⓔ LCD

Figure 5-16
Different types of segmented displays.

LIGHT EMITTING DIODE DISPLAYS

You have already been acquainted with the 7-segment light emitting diode, or LED, display shown in Figure 5-16a. These are currently the most widely used of all display technologies. As you are aware, 7-segment LED displays are available in either a common anode or common cathode configuration. In a common anode configuration, all the segment anodes are tied together and connected to the positive supply, as shown in Figure 5-17a. Conversely, the segment cathodes are tied together and connected to ground in a common cathode configuration, as shown in part b of the figure. In general, a common anode LED requires logic 0's to illuminate the segments, while a common cathode LED uses logic 1's. Popular display colors include red, green, yellow, and orange.

INCANDESCENT DISPLAYS

Each segment of the incandescent display in Figure 5-16b consists of a thin tungsten wire filament that emits a brilliant white light when current is passed through it. Incandescent displays are used where high brightness levels are required, as in direct sunlight. Consequently, they are commonly found in military and avionics applications. A color filter in front of the display can make the light any color desired. Most incandescent displays can be driven directly from open collector TTL output signals.

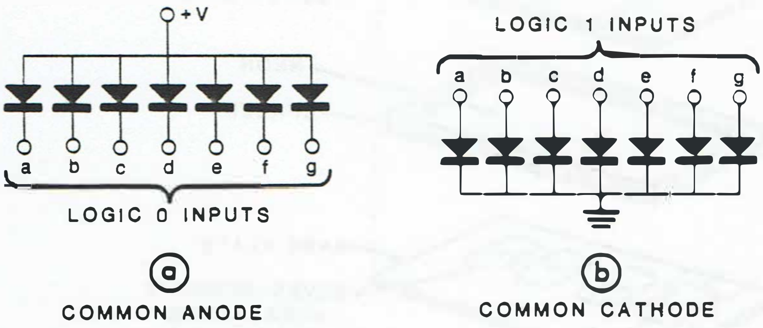


Figure 5-17

Common anode (a) versus common cathode (b) 7-segment LED displays.

GAS DISCHARGE DISPLAYS

The gas discharge display in Figure 5-16c is also used where high brightness levels are required. The construction of a typical four character display is shown in Figure 5-18. Here, a layer of neon gas is sandwiched between a transparent anode plate and a cathode base plate. The top plate is made of clear glass with a transparent conductive anode film deposited on the underside of the glass. The base plate contains the segment pattern. Each deposited segment acts as a separate cathode. Notice that leads are brought-out from each cathode segment.

In operation, a high potential (approximately 200 volts) is applied to the anode film. The various segments are then illuminated by applying a ground, or logic 0, to the cathode segments. This difference in potential between anode and cathode causes the neon gas surrounding a given cathode segment to ionize and emit a red-orange glow.

In most cases, gas discharge displays only come in multicharacter modules as shown. The modules usually contain the necessary decoding and drive circuitry required to operate directly from TTL signals in the BCD format.

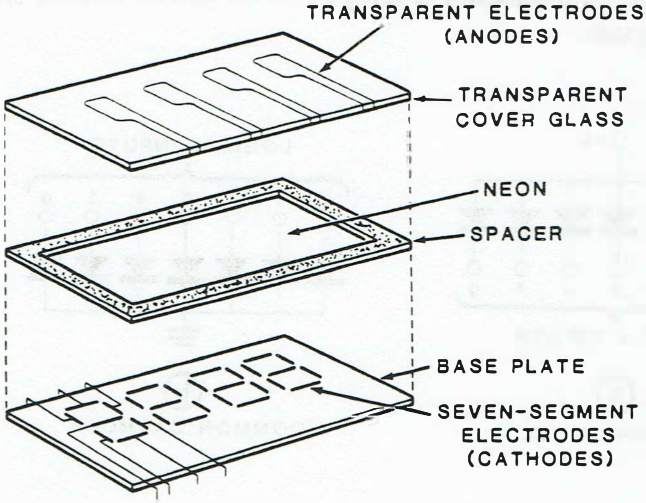

TRANSPARENT ELECTRODES (ANODES)
TRANSPARENT COVER GLASS
NEON
SPACER
BASE PLATE
SEVEN-SEGMENT ELECTRODES (CATHODES)

Figure 5-18

Construction of a gas discharge display.

VACUUM FLUORESCENT DISPLAYS

You will recognize vacuum fluorescent, or VF, displays by the soft green-blue light that they emit. Most VF displays are contained in flat, multicharacter packages as shown in Figure 5-16d.

The construction of a 7-segment VF character is illustrated in Figure 5-19. A conductive interconnect pattern is first printed onto a glass substrate. This pattern is then covered with an insulating film, except for the segment areas. Anode

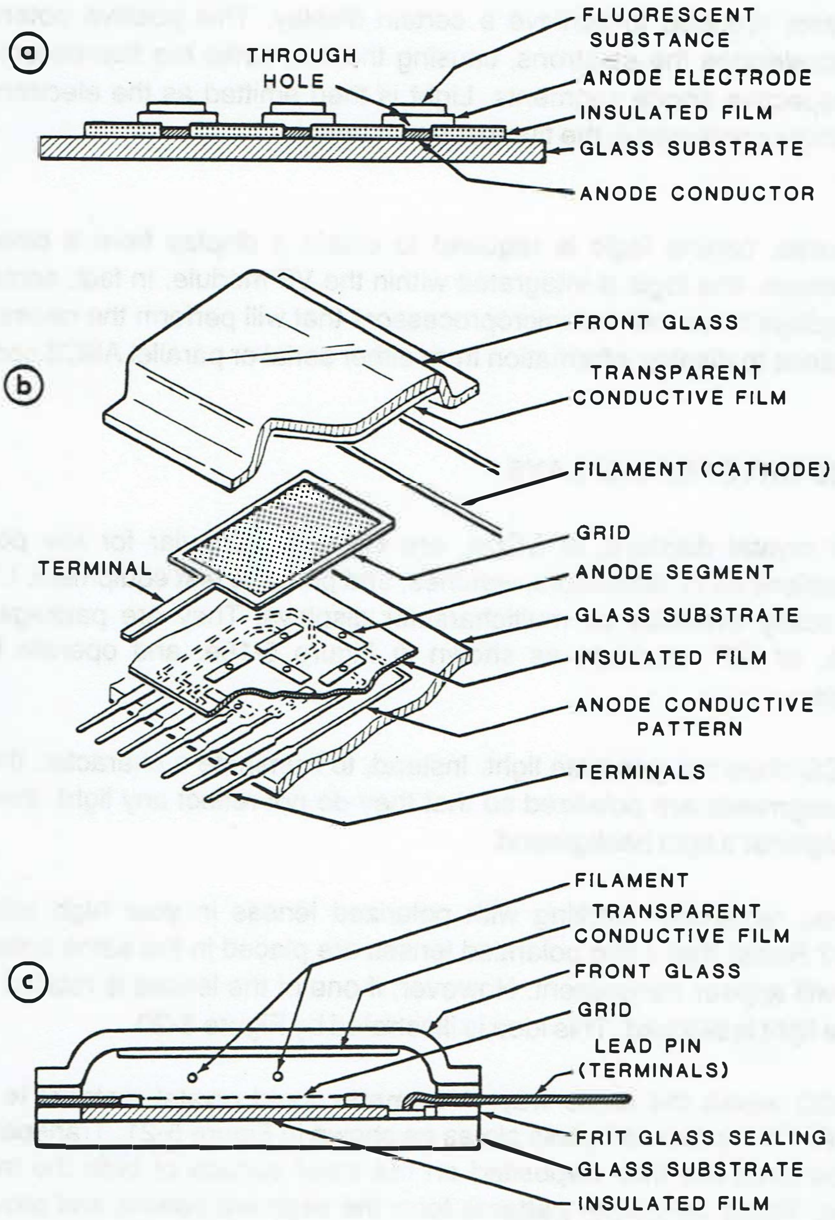


Figure 5-19

Construction of a VF display.

segments are mounted on top of the film and connect to the conductor pattern through the holes in the insulating film as you can see from parts a and b of the figure. Each segment anode is coated with a fluorescent material. A spacer grid is then placed over the anode segments and cathode filament wires are positioned over the grid. The front glass is finally bonded to the assembly to form an air tight glass envelope as shown in Figure 5-19c.

Here's how it works. Approximately 6 volts is applied to the cathode filament wires. This causes the filament to heat-up and give off electrons through thermal emission. A 15- to 30-volt potential is then applied to the grid and those anode segments required to achieve a certain display. This positive potential attracts and accelerates the electrons, causing them to strike the fluorescent coating on the respective anode segments. Light is then emitted as the electrons strike the phosphors contained in the fluorescent material.

Of course, control logic is required to obtain a display from a binary input. In most cases, this logic is integrated within the VF module. In fact, some **multi-line** VF displays have internal microprocessors that will perform the necessary control operations to display information from either serial or parallel ASCII code.

LIQUID CRYSTAL DISPLAYS

Liquid crystal displays, or LCDs, are extremely popular for low power CMOS applications as in calculators, watches, and portable test equipment. LCD displays are usually available as multicharacter displays. They are packaged in a dual in-line, or DIP, package as shown in Figure 5-16e, and operate from CMOS signals.

An LCD does not generate light. Instead, to illuminate a character, the respective LCD segments are polarized so that they do not reflect any light, thereby looking dark against a light background.

Do you remember working with polarized lenses in your high school science class? Recall that if two polarized lenses are placed in the same polarizing plane, they will appear transparent. However, if one of the lenses is rotated 90 degrees, all the light is blocked. This idea is illustrated by Figure 5-20.

An LCD works the same way. A **nematic** liquid crystal material is sandwiched between two polarized glass plates as shown in Figure 5-21. Transparent conductive patterns are then deposited on the inner surface of both the front and rear plates. These conductor patterns form the segment pattern and provide connections to the segments.

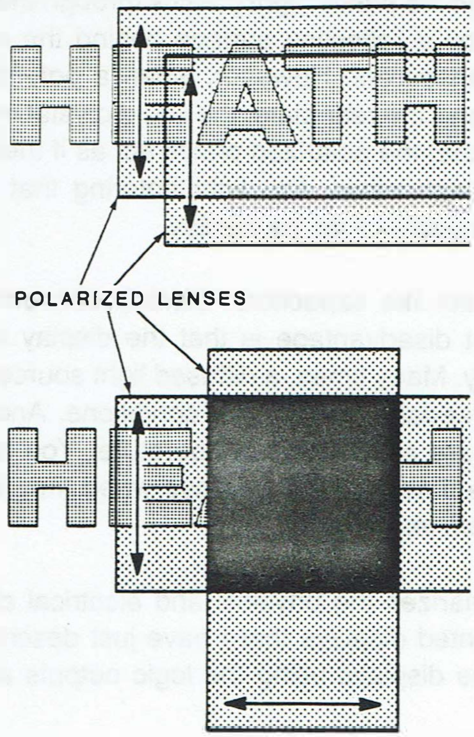


Figure 5-20
The polarization effect.

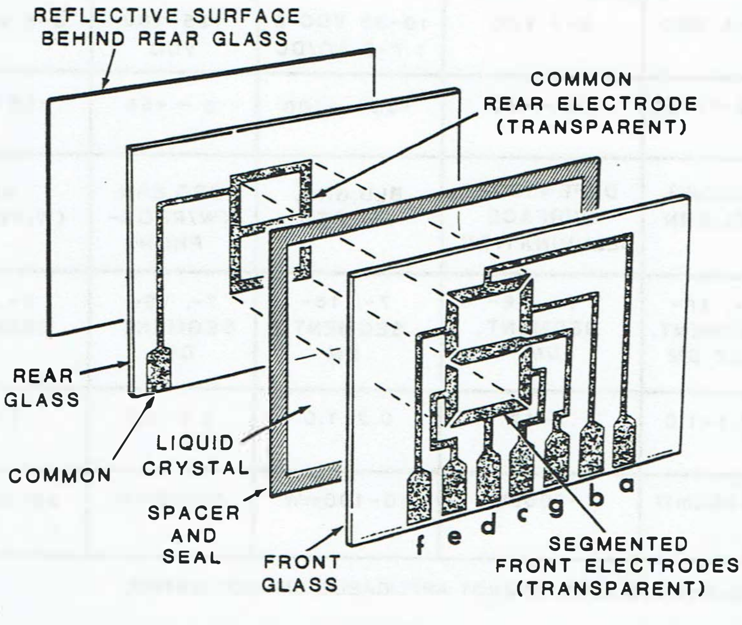


Figure 5-21
Construction of a LCD.

When no potential exists between the plates, light passes through the liquid crystal material and is reflected back by a reflecting surface behind the rear plate. As a result, the segments look transparent. However, when a potential is applied between a front plate segment and the rear plate, the liquid crystal material bends the light under the segment so that the polarized plates act as if their polarization planes are at 90 degrees to each other. Any light entering that segment will be absorbed and result in a dark segment on the display.

The polarized LCD segments act like capacitors, thereby consuming very little power. Of course, their biggest disadvantage is that the display area must be well lit in order to see the display. Many times, a diffused light source is positioned behind the display to make it readable in low light conditions. Another problem with LCD displays is that they are sensitive to temperature. You have probably experienced the slow response time of these displays when they are subjected to temperatures much below room temperature.

The table in Figure 5-22 summarizes the physical and electrical characteristics of the different types of segmented displays that I have just described. Now it's time to learn how to drive these displays using the logic outputs available from digital circuits.

	LED	LCD	VACUUM FLUORESCENT	GAS DISCHARGE	INCANDESCENT
BRIGHTNESS	G TO E	NA	G	G TO E	E
VOLTAGE	2-4 VDC	3-7 VAC	10-35 VDC & 1.7-5 AC/DC	125-180 VDC	3-5 VDC/AC
TEMPERATURE RANGE (°C)	-55 — +100	-15 — +80	-55 — +100	0 — +55	-55 — +80
COLORS	RED, ORG, YEL, GRN	DEPENDS ON SURFACE ILLUMINATION	BLU, GRN, YEL, RED	ORG, GRN, (W/PHOSPHOR)	ALL (W/FILTERS)
DISPLAY STYLE	7-, 16-SEGMENT, 5x7 DM	7-, 16-SEGMENT, DM	7-, 16-SEGMENT, DM	7-, 16-SEGMENT, DM	7-, 16-SEGMENT
CHARACTER HGT (IN.)	0.1-1.0	.2-12	0.2-1.0	0.2-1.0	TO 1
POWER/ CHARACTER	2-250mW	1-100μW	10-100mW	30-90mW	50-1000mW

CODE: G= GOOD; E= EXCELLENT; NA= NOT APPLICABLE; DM= DOT MATRIX

Figure 5-22

A summary of display characteristics.

DECODING AND DRIVING SEGMENTED DISPLAYS

The logic required to illuminate segmented displays is basically the same, regardless of the type of display. The only exception to this rule is LCD displays, which will be discussed separately.

All 7-segment displays require that binary signals be decoded into a 7-segment character code which provides enough signal power to illuminate the segments. This is the job of a device called a **decoder/driver**. For single character displays, the decoder/driver is a separate MSI device that you must design into the display circuit. With multicharacter displays, the decoder/driver is usually contained within the display module or integrated into a display chip.

The first job of the decoder/driver is to convert binary signals to 7-segment code. Recall that you designed such a circuit in Units 2 and 3 using SSI and MSI devices. Fortunately, there are single-chip MSI devices available that accomplish this task. These devices were mentioned briefly in Unit 3, but it is now time to see precisely how they operate with the displays described earlier.

The 74LS47 BCD-to-Seven Segment Decoder/Driver

The 74LS47 generates 0's to illuminate the display segments. Consequently, it is used with common cathode 7-segment LEDs as well as incandescent and gas discharge devices. The 74LS47 has open collector outputs that must be connected to the segments via pull-up resistors as shown in Figure 5-23. A 330 ohm resistor value is typical. When driving incandescent displays, pull-up resistors are often not necessary, since the incandescent filaments provide enough resistance to limit the current flow.

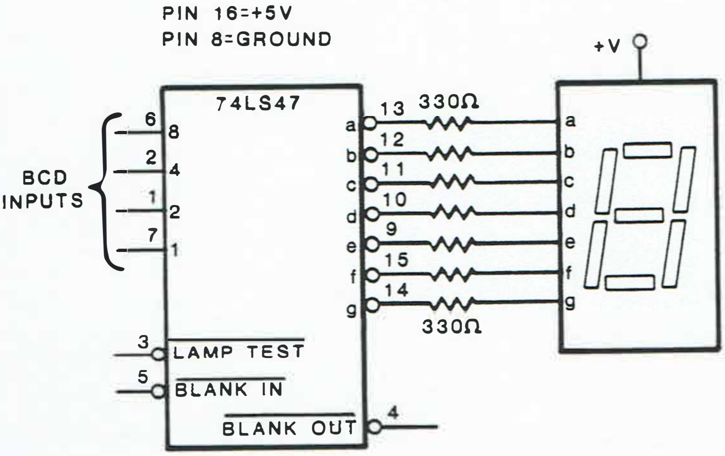


Figure 5-23

A 74LS47 decoder/driver used with a common anode display.

Each segment output of the 74LS47 can sink up to 24 milliamps and be pulled-up to as much as 15 volts. This is adequate to drive most LED displays, but might not do the job for some incandescent displays. I should mention, however, that the 7447 standard TTL version has an output that can sink up to 40 milliamps and be pulled-up to 30 volts. This is adequate for all LED and most incandescent displays.

Notice the two input control lines on the 74LS47, labeled $\overline{\text{LAMP TEST}}$ and $\overline{\text{BLANK IN}}$. As you can tell, both are active low. When $\overline{\text{LAMP TEST}}$ is activated, all the display segments should illuminate. If they don't, the display or the driver is most likely defective. This allows you to test the display circuit.

The $\overline{\text{BLANK IN}}$ control line provides a **ripple blanking** feature for the display. It works like this: Suppose that you wish to display the value 80 using a four character display. Without ripple blanking, the display would generate 0080 rather than 80. The difference is the presence of the leading zeros. The value 80 can be displayed if the two most significant displays are turned completely off, or **blanked**. To do this, the $\overline{\text{BLANK IN}}$ line must be held low. When this line is low, the display is turned off for a BCD input value of 0000.

Now, notice that the 74LS47 also includes a $\overline{\text{BLANK OUT}}$ line. This line generates a logic 0 each time the display is blanked. If the $\overline{\text{BLANK OUT}}$ line of one display is connected to the $\overline{\text{BLANK IN}}$ line of the next display, it too will turn-off for a BCD input of 0000. This idea is illustrated by Figure 5-24. Notice that the most significant 74LS47 has its $\overline{\text{BLANK IN}}$ tied low. The $\overline{\text{BLANK OUT}}$ line of this decoder/driver is then connected to the $\overline{\text{BLANK IN}}$ line of the next 74LS47, and so on. With this arrangement, any leading zeros will be automatically blanked-out of the display. However, any zeros within the number, such as the 0 in 80, will not be blanked, since the preceding 74LS47 will not generate a $\overline{\text{BLANK OUT}}$ signal.

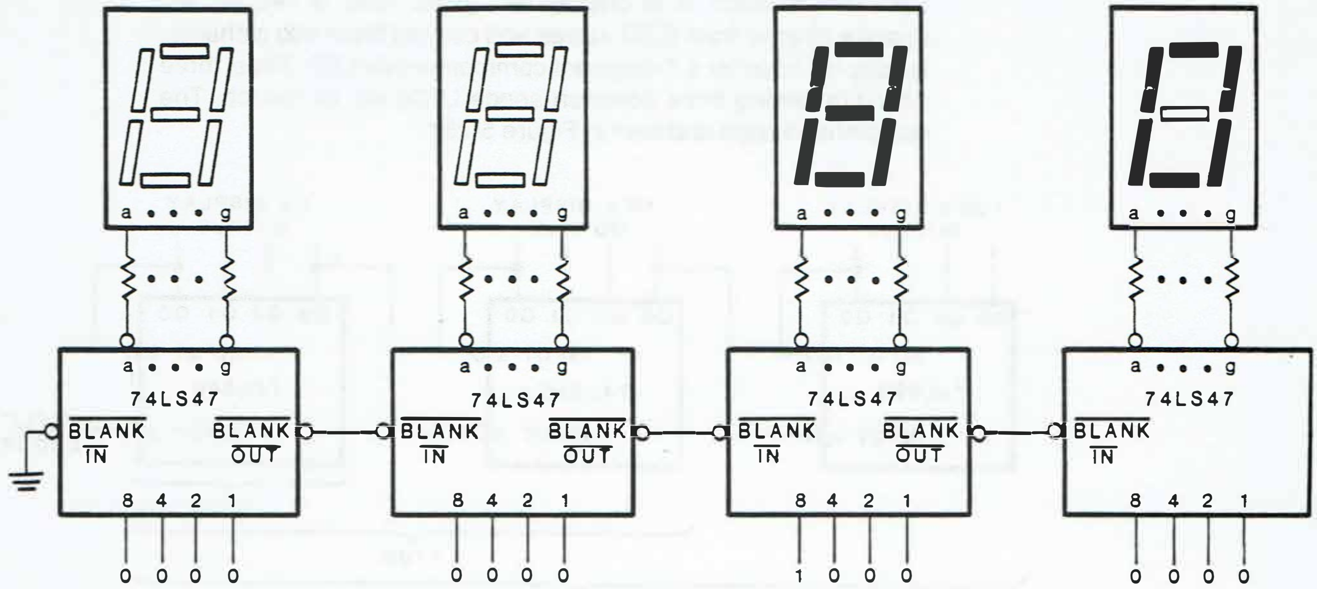


Figure 5-24

The ripple blanking feature of the 74LS47 is used to blank-out leading zeros.

Example 5-3

Design a display circuit that will count from 000 to 999. Be sure to include leading zero blanking in your circuit.

First, the circuit must count from 000 to 999. This is a three digit decimal count where each digit counts from 0 to 9. Are you aware of a counter IC that will generate the required digit count? Of course, the 74LS90 decade counter. So, why not connect, or **cascade**, three 74LS90's together as shown in Figure 5-25a. Here, each counter will generate the count required for each digit of the display. A clock signal is applied to the least significant counter. Then, when this counter rolls over, it will clock the next counter, and so on, to generate a BCD count from 000 to 999.

The next problem is to display the count. Well, a 74LS47 will operate directly from BCD values and convert them into a character display code for a 7-segment common anode LED. Thus, three 74LS47s driving three common anode LEDs will do the job. The completed design is shown in Figure 5-25b.

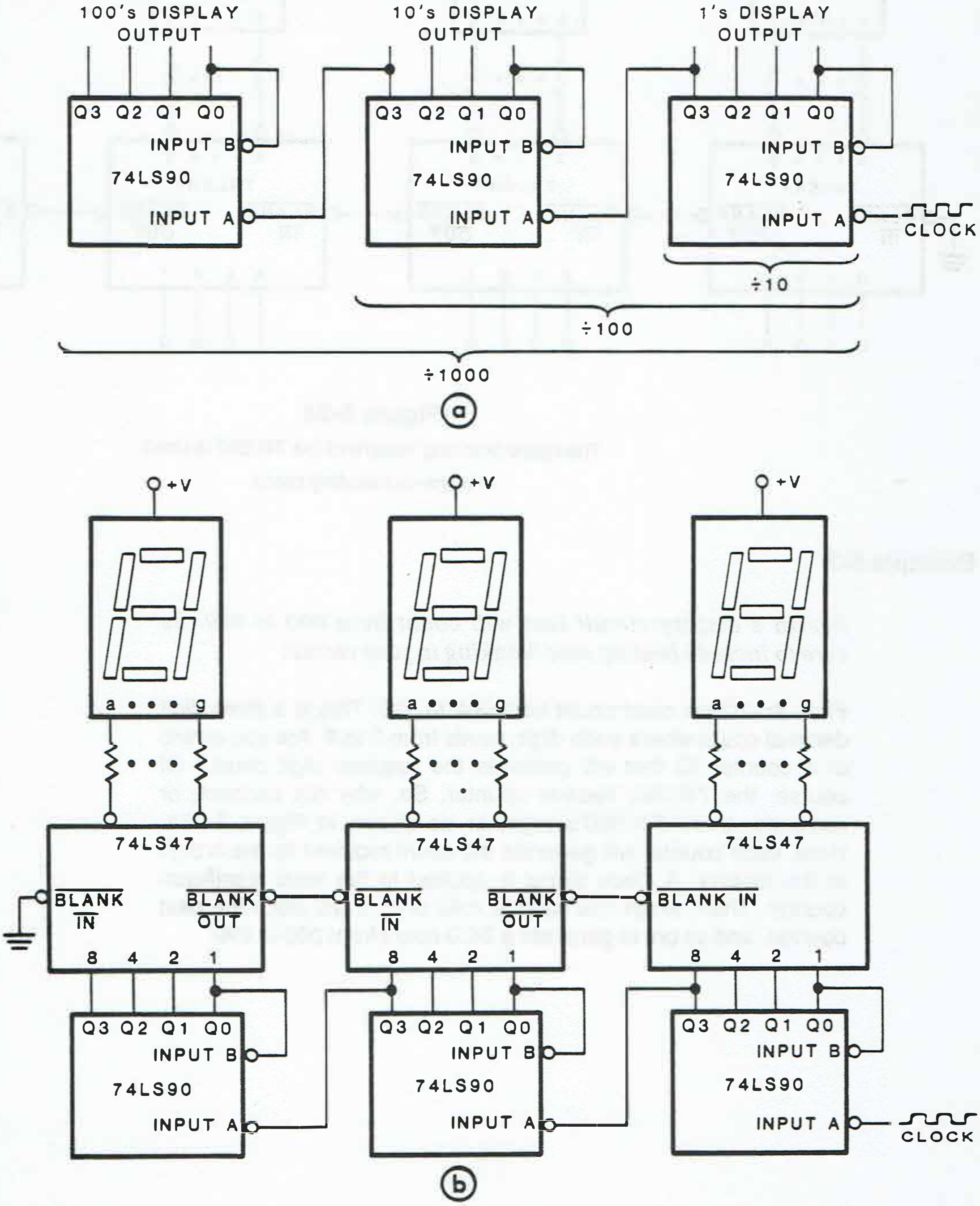


Figure 5-25

A circuit to display a decimal count from 000 to 999 (Example 5-3).

The circuit just designed is used quite often in commercial products to display decimal counts. For example, how about a digital gas pump that counts and displays the dollar amount of gas you are pumping? As another example, consider an electronic frequency counter that counts cycles. In each case, the count is incremented by applying a clock pulse to the least significant counter.

The 74LS48 BCD-to-Seven Segment Decoder/Driver

The 74LS48 generates logic 1's to illuminate display segments. Consequently, it is used with common cathode displays as shown in Figure 5-26. The 74LS48 does not require pull-up resistors and can drive a display directly as shown in Figure 5-26a. However, the standard TTL 7448 version in Figure 5-26b requires that external pull-up resistors be added to each segment line. These resistors are required to limit the amount of current sunk by the device when it generates logic 0's on its output lines. A 330 ohm resistor value is typical. From Figure 5-26, you can also see that both the 74LS48 and 7448 have the same lamp test and blanking features as the 74LS47.

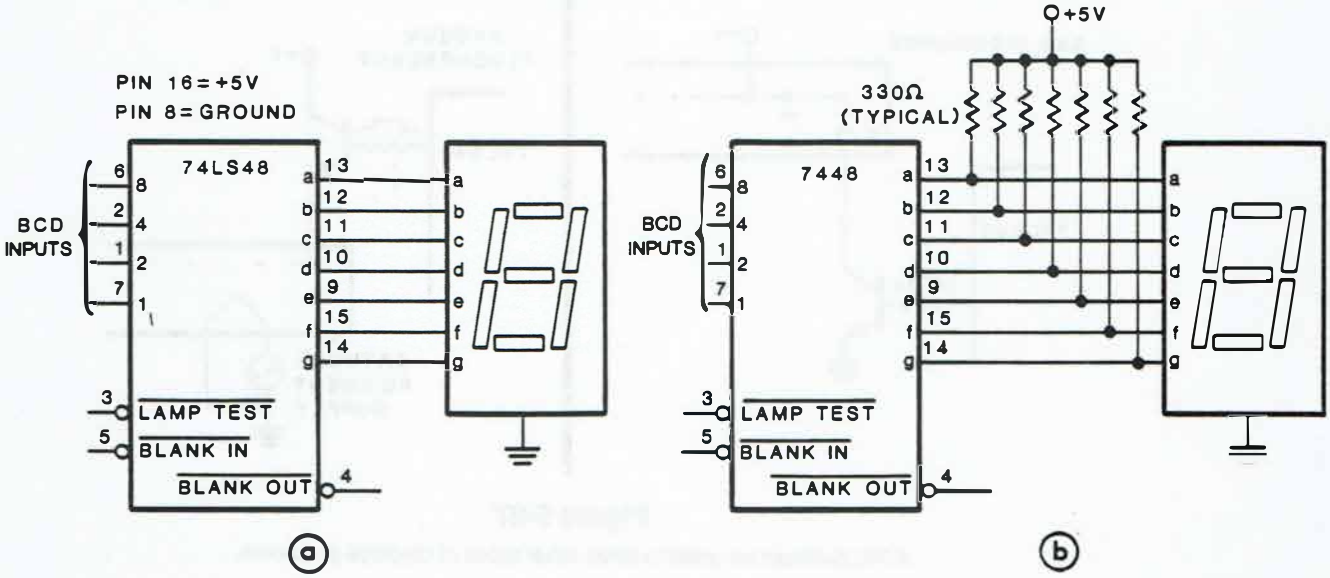


Figure 5-26

A 74LS48 decoder/driver used with a common cathode display (a) and a 7448 decoder/driver (b). The standard TTL 7448 requires external pull-up resistors.

The 74LS48 can also be used to drive other types of displays as shown in Figure 5-27. In each case, the 74LS48 outputs are applied to the base of a **drive transistor**. Think of the transistor as a switch. A logic 1 applied to the transistor turns it on and closes the switch, while a logic 0 turns it off and opens the switch. Thus, a logic 1 output from the 74LS48 turns the transistor on and completes the circuit to ground for a given segment. Drive transistors are used, since the 74LS48 cannot source enough current to drive these displays directly.

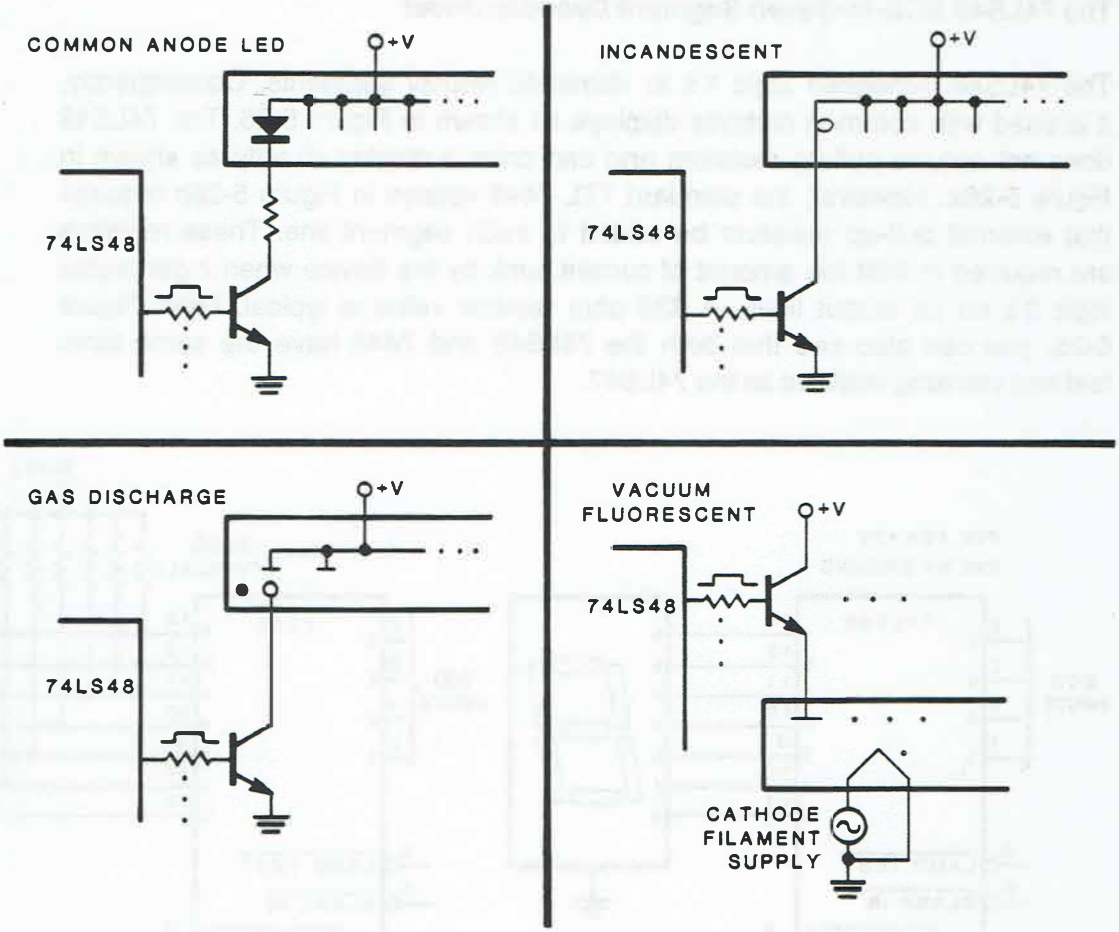


Figure 5-27

A 74LS48 can be used to drive other types of displays as shown.

The CD4511/14511 CMOS Decoder/Driver

This device, shown in Figure 5-28, is used to drive 7-segment displays from CMOS circuits. It generates a logic 1 level to illuminate a common cathode LED display. A series resistor is used to limit the current through the display or to adjust the display brightness. The resistor value depends on the CD4511 supply voltage and the current limitations of the LED display.

The CD4511/14511 includes the lamp test and blanking input features like the earlier TTL decoder/drivers. In addition, a latching feature is provided to hold a given display. When a logic 1 is applied to the LATCH input line the display will hold the present display character.

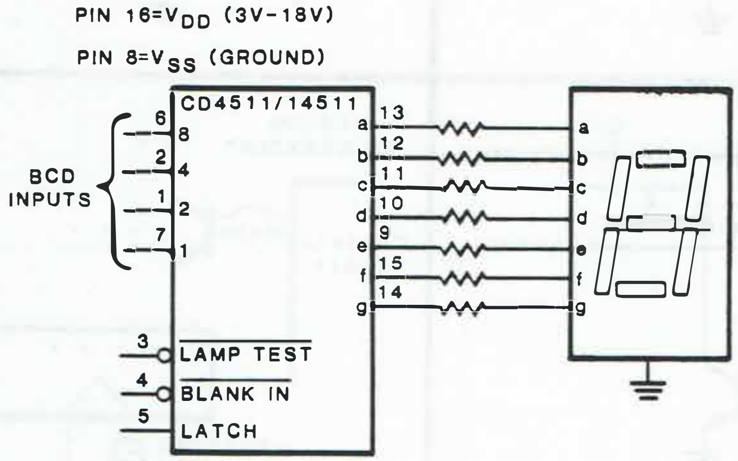


Figure 5-28

A CD4511/14511 decoder driver is used to drive 7-segment displays from CMOS circuits.

Other types of displays can also be driven by this device as shown in Figure 5-29. Here, drive transistors are again used to switch-on the respective display segments.

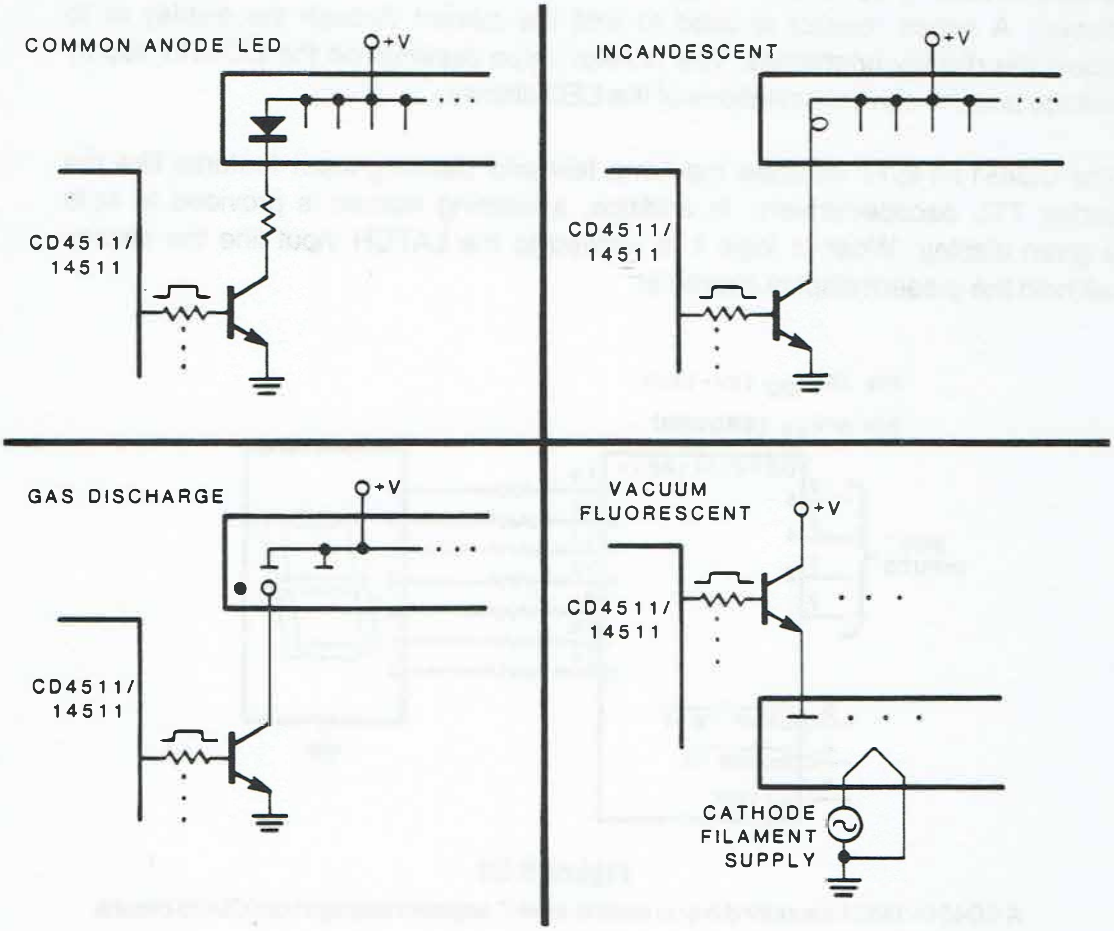


Figure 5-29

A CD4511/14511 can be used to drive other types of displays as shown.

DECODING AND DRIVING LCDs

As you have learned, the construction of LCDs is quite different from other segmented displays. This also holds true of the electrical requirements of LCDs. First, only AC signals can be applied to the LCD segments, not DC. Any DC applied to a segment will shorten the life of the display. In addition, the AC signal must be perfectly symmetrical around zero so that no DC component is introduced. Both sine and square wave signals are acceptable, but if a square wave signal is used, it must have a 50% duty cycle, or a DC offset will be introduced.

Since the LCD segments act like capacitors, the frequency of the AC signal must be kept low. This avoids current drain due to capacitive coupling of the signal. (In any capacitive circuit, impedance, or resistance to current flow, decreases as frequency goes up.) Because very low power consumption is a major advantage of the LCD, the drive frequency should be kept low. On the other hand, at very low frequencies, flicker will develop as the display turns on and off with each cycle of the drive signal. For this reason, drive frequencies between 30 Hz and 40 HZ are common.

The LCD is driven by applying an AC drive signal to both the backplane electrode and to all segment electrodes. To turn a given segment on, the phase of the drive signal for that segment is shifted by exactly 180 degrees from the phase of the backplane signal. Figure 5-30 uses square wave signals to show that when the signals are in phase, there is no potential difference developed across the segment and the backplane electrodes. However, when the signals are 180 degrees out-of-phase, an absolute potential difference is created between the segment and backplane. This potential difference produces a square wave which is centered around zero and whose positive and negative peaks equal the two component signal levels. This results in a darkened segment due to the polarization effect discussed earlier.

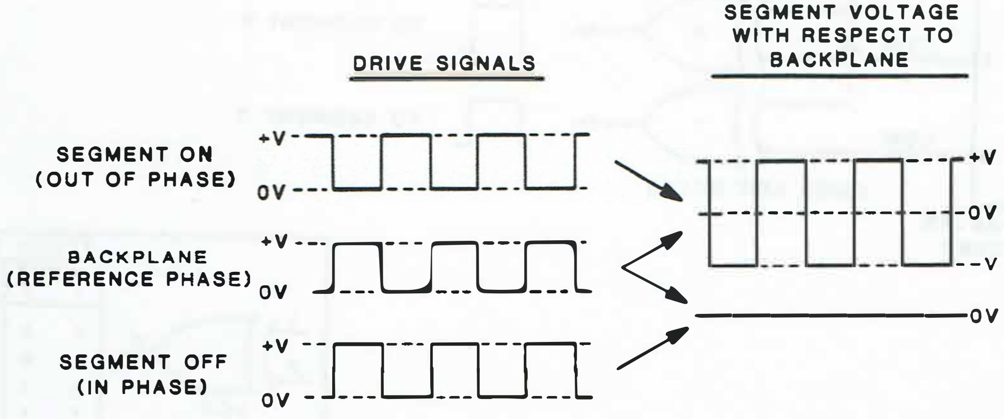
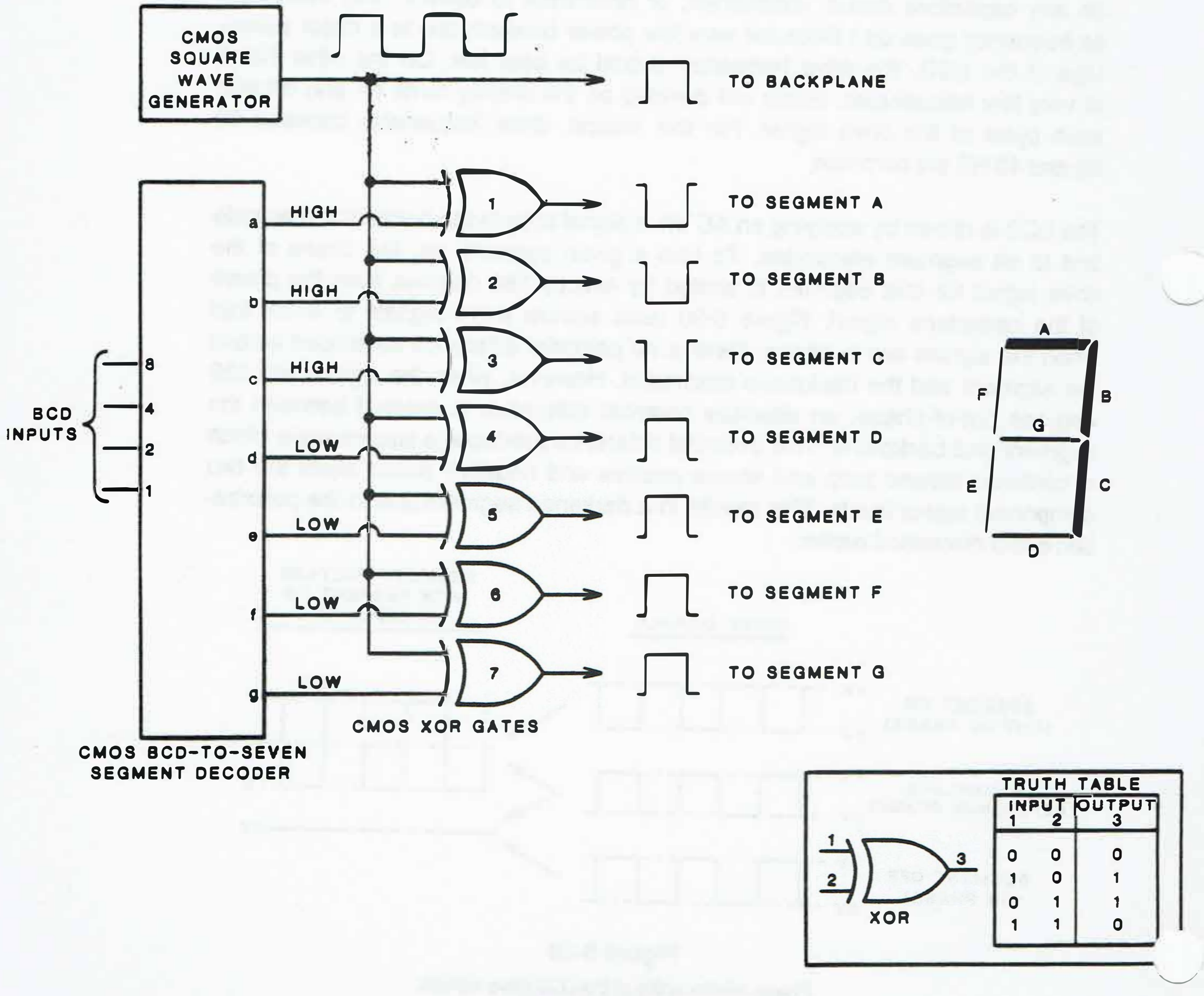

DRIVE SIGNALS
SEGMENT VOLTAGE WITH RESPECT TO BACKPLANE
SEGMENT ON (OUT OF PHASE)
+V
0V
BACKPLANE (REFERENCE PHASE)
+V
0V
SEGMENT OFF (IN PHASE)
+V
0V
+V
0V
-V
0V

Figure 5-30

Phase relationships of the LCD drive signals.

Now, what type of gate do you suppose is used to generate the out-of-phase condition? (Hint: What gate can be used to detect out-of-phase conditions?) You're right, an XOR gate! Most LCD drivers use XOR gates connected to a standard decoder/driver as shown in Figure 5-31. A square wave generator supplies the reference signal to the LCD backplane and one side of an XOR gate for each LCD segment. The other side of each XOR gate is driven by the output of a BCD-to-seven segment decoder that has active high outputs. As long as the decoder outputs are low, the XOR gate output is in-phase with the backplane reference signal. However, when a given decoder output goes high, its associated XOR gate inverts the drive signal to produce the 180 degree phase shift. Notice that the decoder outputs shown in Figure 5-31 will display the number 7.



TRUTH TABLE

INPUT 1	INPUT 2	OUTPUT 3
0	0	0
1	0	1
0	1	1
1	1	0

Figure 5-31

An LCD decoder/driver circuit.

The CD4543/14543 CMOS LCD Decoder/Driver

The CD4543 and 14543 are pin-for-pin compatible BCD-to-seven segment decoder/drivers used for LCD displays. A typical drive circuit is shown in Figure 5-32. A square wave signal must be applied simultaneously to the PHASE input of the decoder/driver and the backplane connection of the display. The inputs to the decoder/driver are the standard 8-4-2-1 BCD code and the seven decoder/driver outputs are connected directly to the segment lines of the display.

Notice that the CD4543/14543 includes both a blanking and latch feature. The display will be blanked when a logic 1 is applied to the BLANK input line. When a logic 0 is applied to the $\overline{\text{LATCH}}$ input line, the display will hold the present display value.

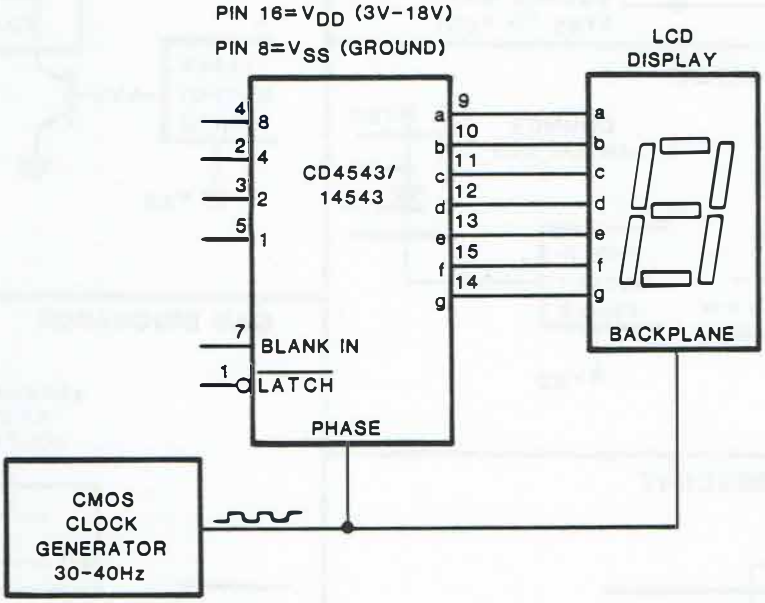


Figure 5-32

The CD4543/14543 CMOS decoder/driver is used for 7-segment LCD displays.

The CD4543/14543 is so versatile that it can be used to drive any type of 7-segment displays that you learned about earlier. The respective connections for LED, incandescent, gas discharge, vacuum fluorescent, and liquid crystal displays are shown in Figure 5-33. Transistors are required to drive incandescent and gas discharge displays, since the decoder/driver is not capable of sourcing enough current to drive these displays.

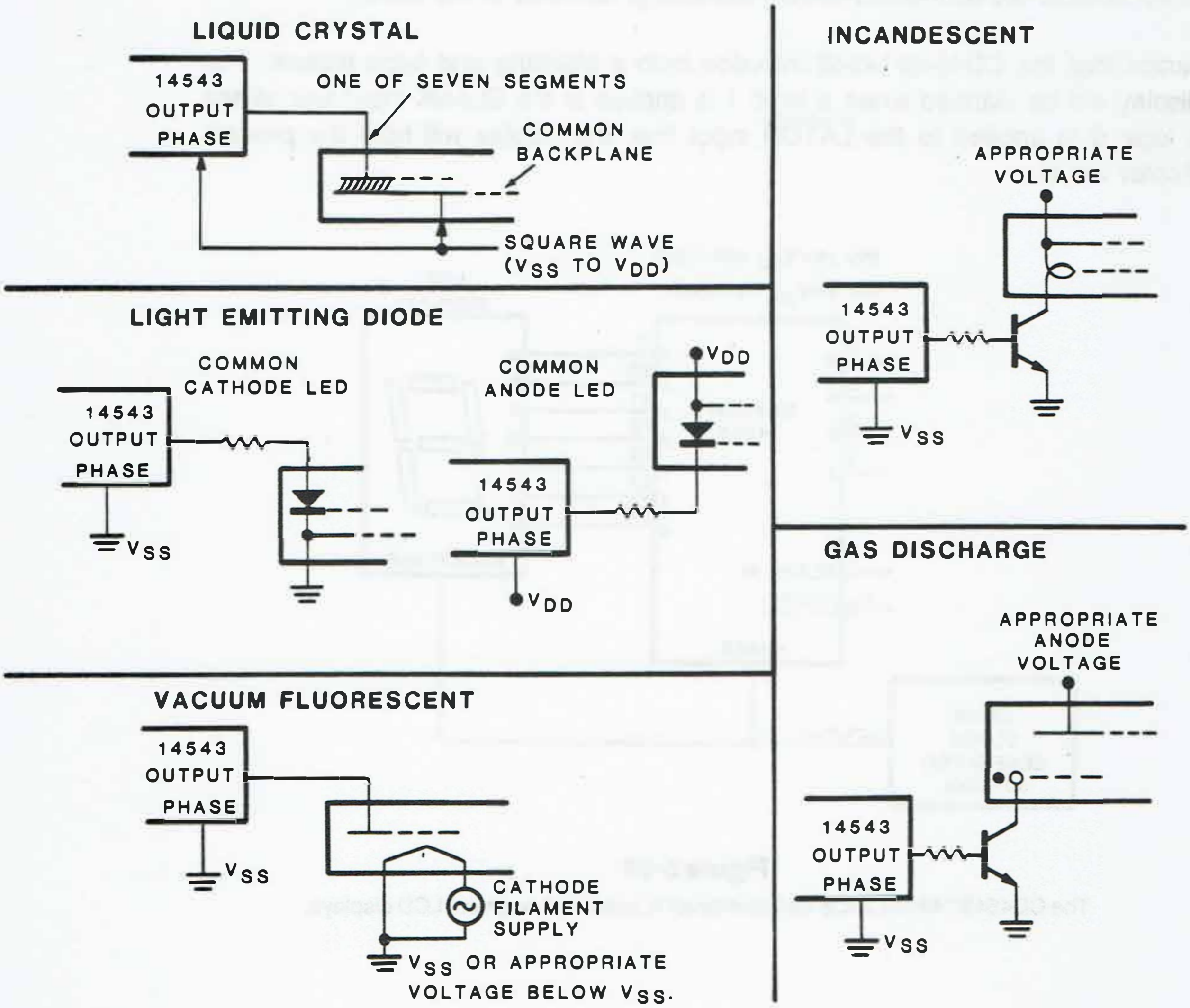


Figure 5-33

The CD4543/14543 can be used to drive all types of displays as shown.

MULTIPLEXED DISPLAYS

Go back and look at the circuit you designed in Example 5-3 (Figure 5-25b). The problem with this circuit is that a separate decoder/driver is required for each digit in the count. You can reduce the number of decoder/drivers to **one** for any number of display digits using a technique called **display multiplexing**.

Here's the idea. Suppose you install two four-position switches in the circuit as shown in Figure 5-34. If both switches are initially placed in position 1, the circuit will display the count of counter 1 on display 1. Next, move the switches to position 2. This time, display 1 is turned-off and the count of counter 2 is displayed by display 2. The switches are then moved to position 3 to display the count of counter 3, and so on. Each counter output and respective display is sequentially scanned from right to left in this manner. If the scan rate is fast enough, it will "appear" that all the displays are constantly on, because of the image retention ability of your eyes.

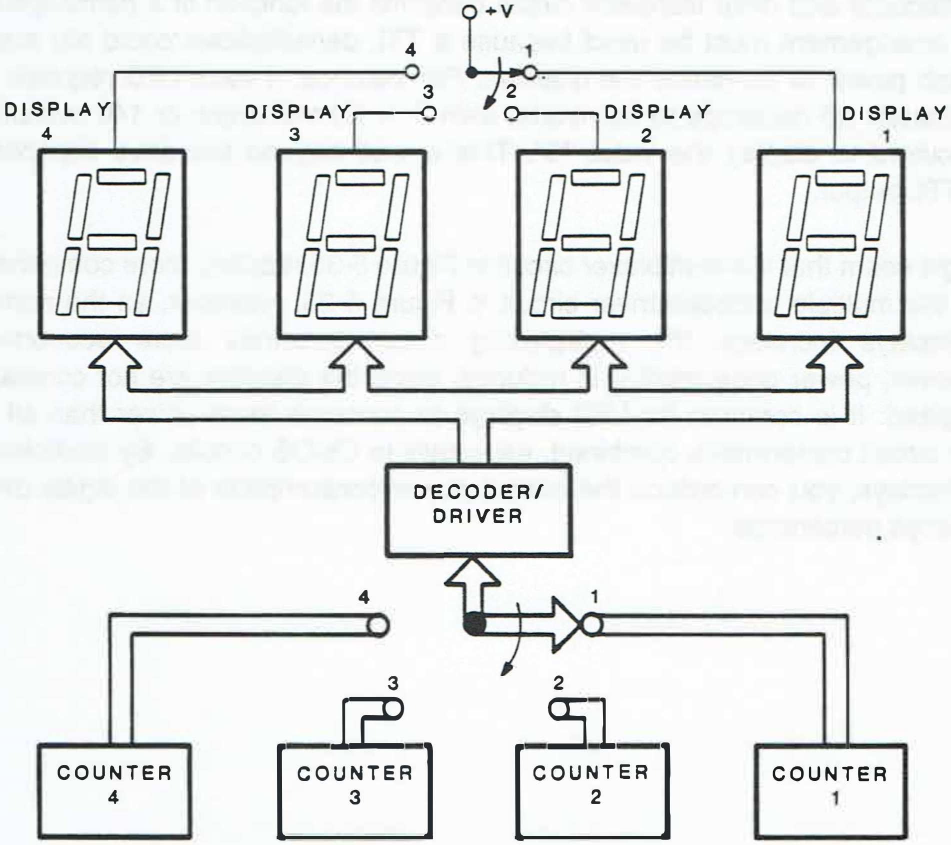


Figure 5-34
A simplified display multiplexing circuit.

As you are aware, the electronic equivalents of a mechanical switch are the multiplexer and the demultiplexer. Notice in Figure 5-34 that the lower switch must **select** and apply one of four counts to the decoder/driver input. This fits the definition of a multiplexer. On the other hand, the upper switch must **distribute** the supply potential to the displays. This fits the definition of a demultiplexer. So, let's replace the bottom switch with a quad 4-to-1 multiplexer and the upper switch with a 1-of-4 demultiplexer.

The resulting display multiplexing circuit is shown in Figure 5-35. Here, a divide-by-4 counter is used to simultaneously select the multiplexer and demultiplexer channels. A 60 Hz counter clock signal is fast enough to prevent flickering. The divide-by-4 counter can be constructed using the two JK flip-flops in a 74LS76 IC. The JK counter output is applied to a quad 4-to-1 multiplexer (two 74LS157s) that passes the respective BCD count to the single 74LS47 decoder/driver. At the same time, the 1-of-4 decoder output turns on the proper drive transistor and applies power to the respective display.

The decoder and drive transistor circuit performs the function of a demultiplexer. This arrangement must be used because a TTL demultiplexer could not supply enough power to illuminate the displays. For instance, if each LED requires approximately 20 milliamps to illuminate, then 7 × 20 milliamps, or 140 milliamps, is required to display the value "8". This is well beyond the drive capabilities of a TTL output.

It might seem that the multiplexer circuit in Figure 5-35 requires more components than the multiple decoder/driver circuit in Figure 5-25. However, as the number of displays increase, the multiplexing circuit becomes more economical. Moreover, power consumption is reduced, since the displays are not constantly energized. It is common for LED displays to consume more power than all the other circuit components combined, especially in CMOS circuits. By multiplexing the displays, you can reduce the overall power consumption of the digital circuit by a large percentage.

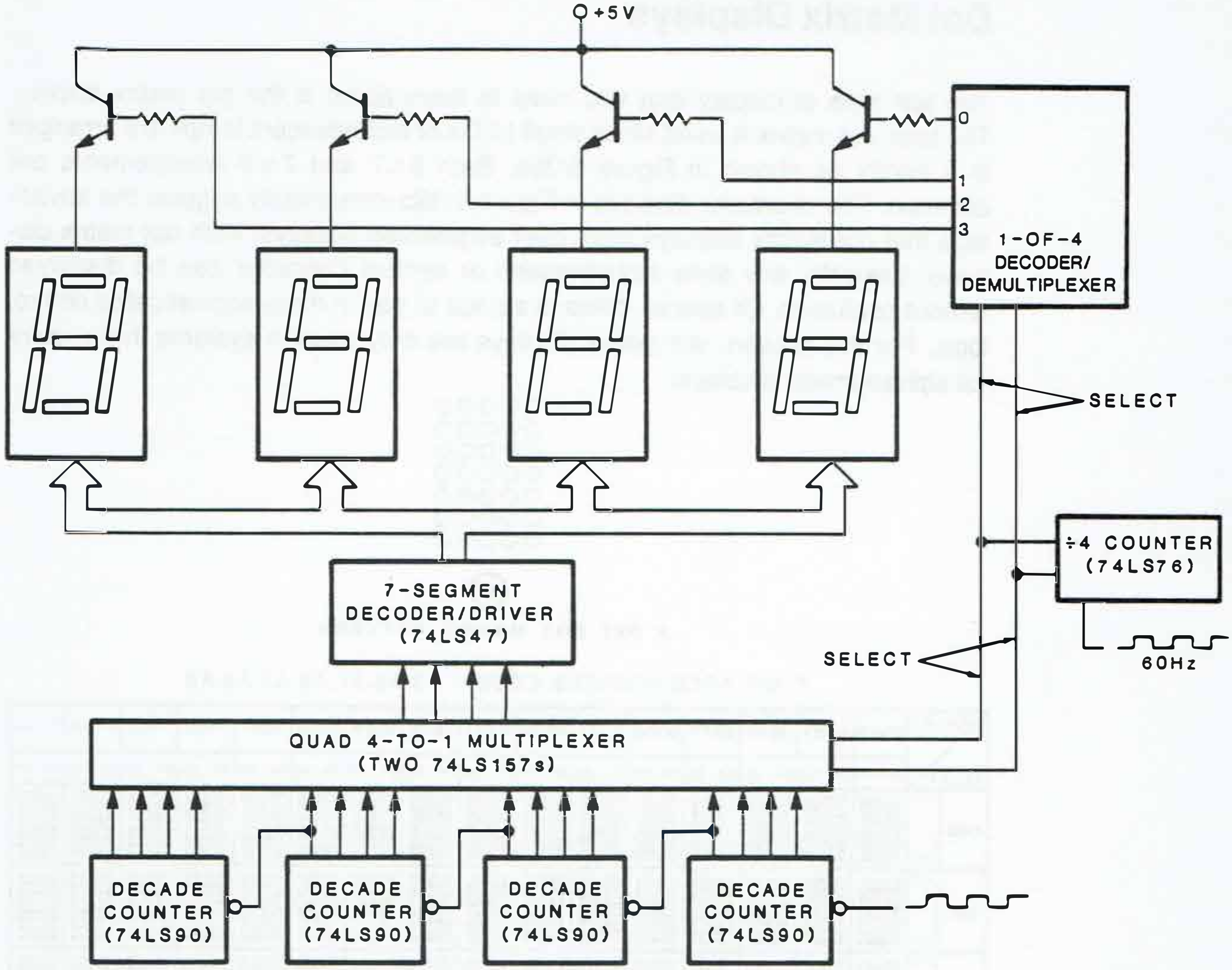
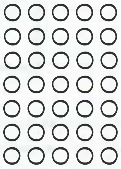


Figure 5-35

A 4-digit multiplexed display.

Dot Matrix Displays

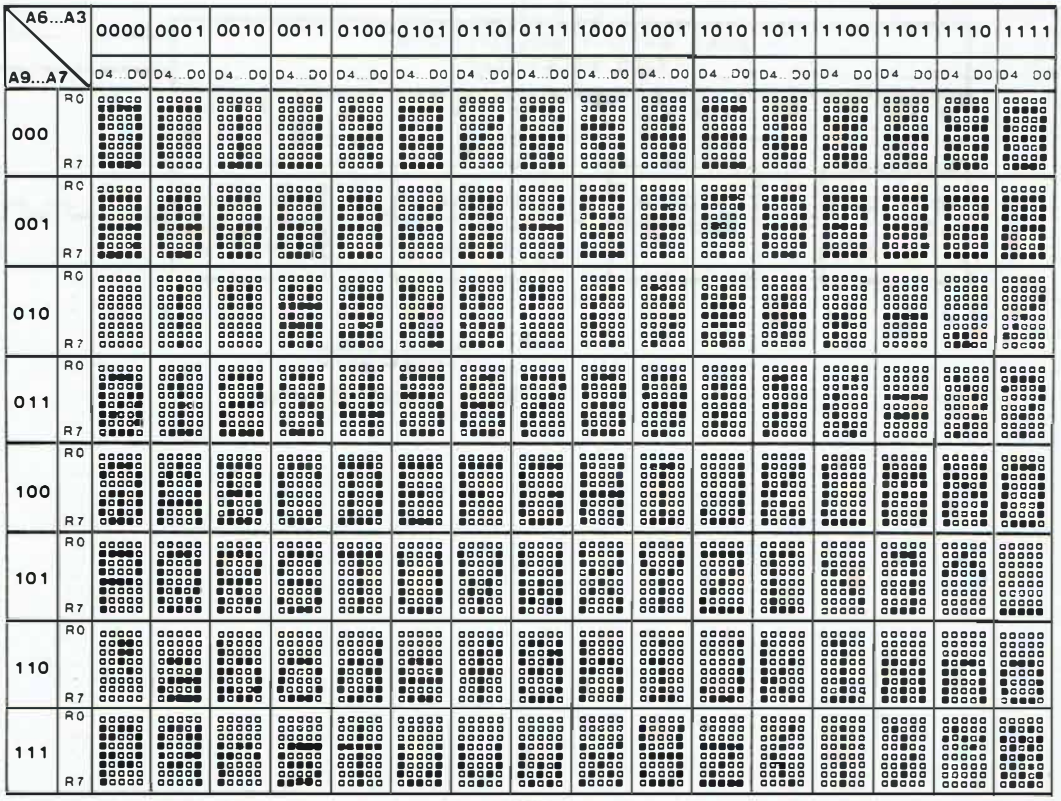
The last type of display that you need to learn about is the dot matrix display. The term dot matrix is used since small LEDs or incandescent lamps are arranged in a matrix as shown in Figure 5-36a. Both 5×7 and 7×9 arrangements are common. The character displays in Figure 5-36b immediately suggest the advantage that dot matrix displays have over segmented displays. With dot matrix displays, basically any style alphanumeric or symbol character can be displayed without confusion. Of course, there is a price to pay in more sophisticated control logic. For this reason, dot matrix displays are only used in systems that require full alphanumeric displays.



ⓐ

A 5x7 DOT MATRIX PATTERN

7-BIT ASCII ADDRESS CODE = A9,A8,A7,A6,A5,A4,A3



ⓑ

TYPICAL DOT MATRIX CHARACTERS

Figure 5-36

Dot matrix displays are more flexible than segmented displays.

CONTROLLING AND DRIVING DOT MATRIX DISPLAYS

A typical 5 × 7 LED dot matrix is shown in Figure 5-37. Notice that it consists of 5 columns and 7 rows of individual LEDs. There are 35 LEDs that connect the columns to the rows. A dot matrix display is always multiplexed, because of the large number of display elements in the matrix. A scanning method, similar to keyboard scanning, is used to display characters within the matrix.

The number of unique displays possible is sufficient to warrant the use of a ROM as the control device. The ROM is called a **character generator ROM**, since it accepts a 7-bit ASCII input and generates the logic required to display the given ASCII character. With a 7-bit ASCII input code, $2^7 = 128$ characters are possible.

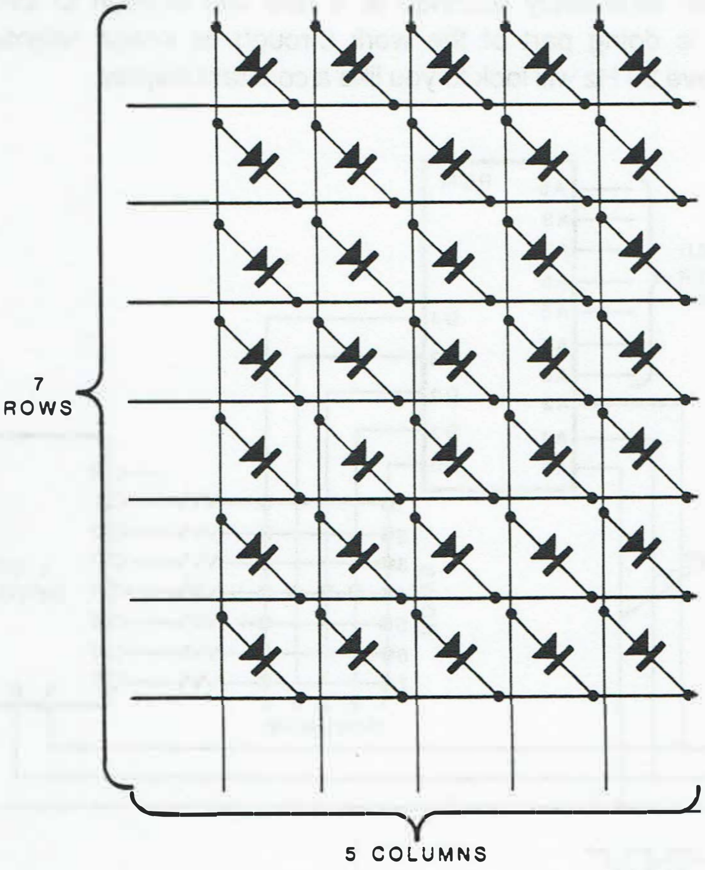


Figure 5-37

A typical 5x7 dot matrix uses LEDs to connect the rows and columns.

A typical dot matrix control circuit is shown in Figure 5-38. The major components are the character generator ROM, a counter, and a decoder. Here's how it works. The decoder will scan the matrix by applying a logic 0 to each row, one at a time. This effectively grounds all the LED cathodes in a given row. The ROM must then generate logic 1's to illuminate only those LEDs within the activated row that are required to display a given character.

For example, Figure 5-38 shows the letter "H" being displayed. The decoder first generates a logic 0 for row 1. At the same time, the ROM must apply a logic 1 to column 1 and column 5. Thus, the ROM output must be 10001. This illuminates the LEDs at the junction of row 1 column 1, and row 1 column 5. Then, the decoder applies a logic 0 to row 2. The ROM output must again be 10001, since the same logic is required to illuminate the letter H for this row. The process continues in this manner until all the rows have been scanned. The matrix is then repeatedly scanned at a rate fast enough to avoid flicker. Again, your eye is doing part of the work through its image retention ability. Any scan rate above 30 Hz will look to you like a constant display.

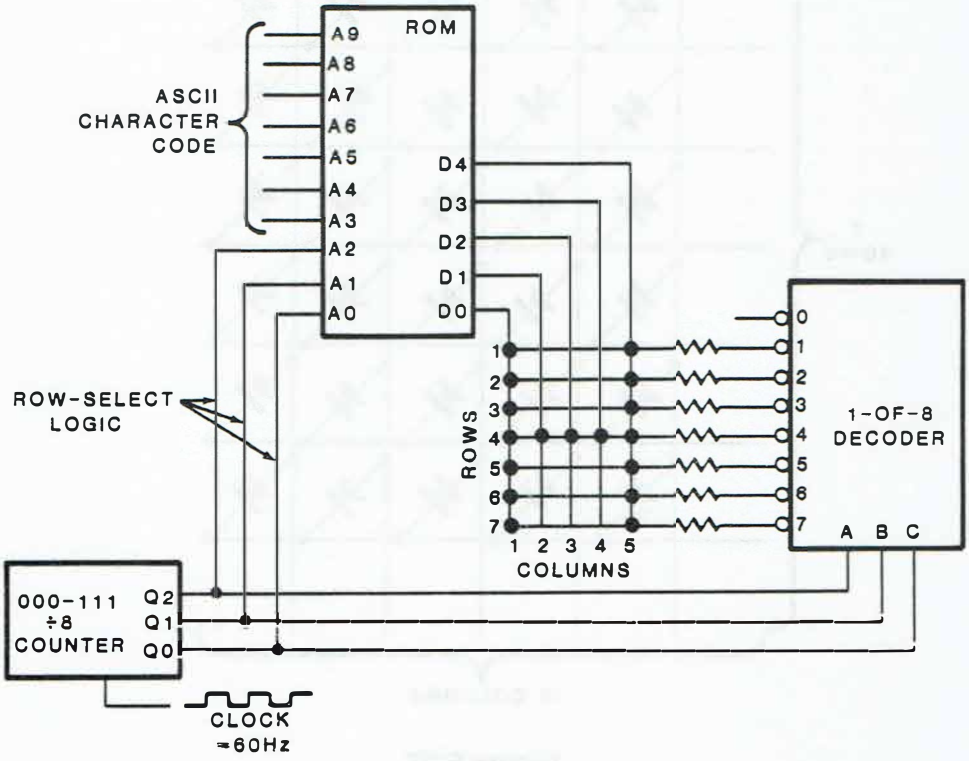


Figure 5-38

A typical diode matrix control circuit.

The counter in Figure 5-38 generates a count from 000 to 111 that does two things:

1. The count is applied to the decoder inputs for row selection.
2. The count is applied to the three least significant ROM address inputs, so that the sequential LED dot code is generated to the matrix rows.

The sequential count applied to the decoder input lines for row scanning should be self-explanatory. Each count causes the decoder to apply a logic 0 to each row in succession. The count is also applied to the three least significant address lines of the ROM. This count is sometimes called the **row-select code**. For each of the 128 possible characters, there must be seven ROM storage locations, one for each set of row data required to illuminate the LEDs. These seven memory locations are assigned to seven 10-bit ROM **addresses**. The 10-bit addresses are made up of the 7-bit ASCII character code plus the 3-bit row-select code provided by the counter. The 7-bit character code is applied to the seven most significant ROM address lines, while the 3-bit row-select code is applied to the three least significant ROM address lines.

For example, the 7-bit ASCII code for the letter H is 1001000. Notice that the letter "H" is found at this address in Figure 5-36. Inside the character generator ROM, there are seven memory locations to represent this character, one for each row. They are at addresses 1001000001 through 1001000111. The contents of each of these locations is the 5-bit logic required to illuminate the five LEDs in a given row. As a result, the letter H would be represented in the ROM as follows:

10-bit ROM Address Input		5-bit ROM Output
7-bit ASCII Code	3-bit Row-select Code	Row Data
1001000	001	10001
1001000	010	10001
1001000	011	10001
1001000	100	11111
1001000	101	10001
1001000	110	10001
1001000	111	10001

Here, the seven most significant address bits are provided by the ASCII code for the letter H, and the three least significant row-select address bits are generated by the counter. Notice that the ROM output is the 5-bit data code required for each row to display the letter H.

You are probably wondering what happens for a row-select code of 000. Notice that output 0 of the decoder is not used. Thus, no row is selected for this count. Since no row is selected, it doesn't matter what is contained in ROM address 1001000000 for the letter H. You could say that this memory location is wasted. In fact, there will be one wasted ROM location for each display character.

Example 5-4

What size ROM is required to implement the 128 characters shown in Figure 5-36b?

The ROM must have ten address lines: seven to select one of 128 characters, and three to generate the row-select code. Since there are five LEDs in each row, the ROM must have at least five data lines to generate the LED row code. Thus, the ROM must be at least a $2^{10} \times 5$, or 1K × 5, ROM.

One final point about the control circuit in Figure 5-38. A latch is required to hold the 7-bit character code at the ROM address input lines until a new character is available. In addition, IC buffers and/or transistor drivers are often used on the ROM and decoder output lines to provide enough power to illuminate the LEDs.

Self-Test Review

11. Open collector drivers are rated by two characteristics. They are:

A. ______________________________

B. ______________________________

12. List the five popular types of 7-segment displays.

A. ______________________________

B. ______________________________

C. ______________________________

D. ______________________________

E. ______________________________

13. Explain how to eliminate leading zeros in a multidigit 7-segment display.

____

____

____

14. Why are drive transistors often required in display circuits? ________

____

____

15. How is an LCD display segment turned-on? ______________

____

____

16. Briefly describe the operation of a multiplexed display. ____________
____
____

17. Alphanumeric displays are generally provided using a __________ __________ type of display.

18. The three major components of a dot matrix display control circuit are a character generator ______, a __________, and a _________.

19. Using the character table in Figure 5-36b, determine the seven character generator ROM addresses that contain the display codes for the letter "S".

20. Assuming a common cathode display matrix, determine the contents of the ROM addresses in question 19.

Answers

11A. maximum current sinking capabilities

11B. maximum pull-up voltage potential

12A. LED

12B. incandescent

12C. gas discharge

12D. vacuum fluorescent

12E. LCD

13. Leading zeros are eliminated using the ripple blanking feature of the display decoder/drivers. The ripple blanking input of the most significant decoder/driver is tied in its active state. The blanking output of this decoder/driver is then connected to the blanking input of the next decoder/driver, and so on.

14. Because the digital devices are not capable of sourcing or sinking enough current to illuminate the display.

15. By shifting the phase of the segment drive signal exactly 180 degrees from the phase of the backplane signal.

Answers (continued)

16. In display multiplexing, each 7-segment display is sequentially turned on via a demultiplexer circuit. At the same time, a multiplexer circuit sequentially supplies the respective display codes.

17. dot matrix

18. ROM, decoder, counter

19. The seven ROM addresses are:

1010011 001
1010011 010
1010011 011
1010011 100
1010011 101
1010011 110
1010011 111

20. The contents of the ROM addresses in question 19 are:

1010011 001 : 01110
1010011 010 : 10001
1010011 011 : 10000
1010011 100 : 01110
1010011 101 : 00001
1010011 110 : 10001
1010011 111 : 01110

Before you read the next section, go to Unit 8 and perform Experiment 14.

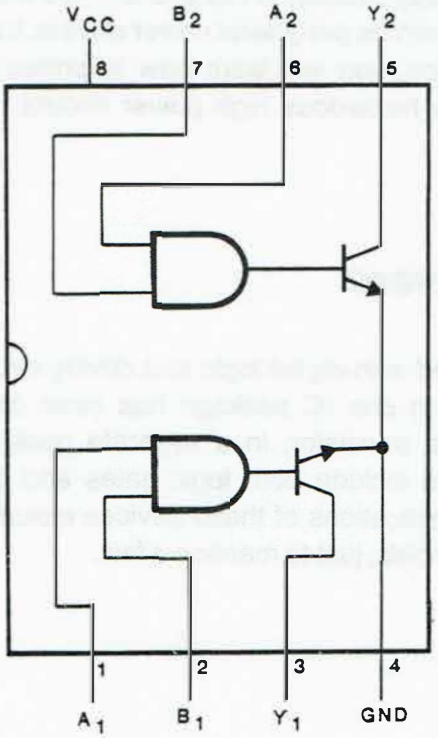
CONTROL DEVICES

In many industrial and commercial applications, digital signals are used to control high power loads. Since digital devices can only source or sink a limited amount of power, some sort of control device must be used between the digital circuit and high power loads. You have already seen the need for these devices, like the driver transistors used in display circuits. In this section, you will be acquainted with additional control devices, such as peripheral power drivers, transistor arrays, and solid state relays. In addition, you will learn how to protect digital circuits and their human operators from hazardous high power circuits using optoelectronic components.

Peripheral Power Drivers

Up to this point, you have worked with digital logic and driving circuits separately. The output of a digital device in one IC package has been connected to the input of a driving device, like a transistor, in a separate package. **Peripheral power drivers** are devices that include both logic gates and drive transistors within the same IC package. Applications of these devices include driving incandescent lamps, relays, and solenoids, just to mention a few.

A typical peripheral driver, the 75461, is shown in Figure 5-39. This device is called a **dual peripheral driver**, since two separate driver circuits are integrated into the same IC package. Each driver circuit consists of a 2-bit AND gate input and a power transistor output. The AND gate inputs are TTL compatible and the driver transistors provide open collector outputs. The output transistors can sink up to 300 milliamps and be used with a pull-up voltage of 30 volts (max).



TOP VIEW

Figure 5-39

The 75461 peripheral power driver contains two AND/driver transistor circuits in one IC package.

A typical relay control application of the 75461 is shown in Figure 5-40. Here, the 75461 peripheral driver is connected between a logic circuit and two 24-volt relays. One side of each AND gate is tied high. One or both relays can then be activated by applying a logic 1 to the other side of the respective AND gate.

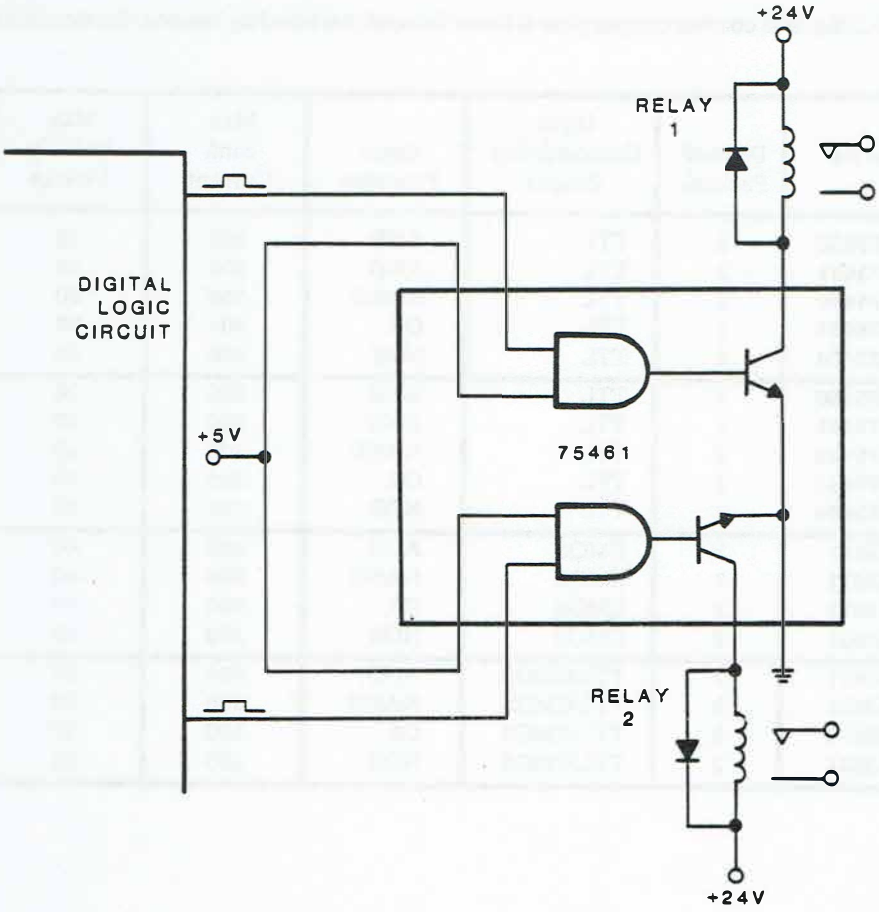


Figure 5-40

A dual relay drive application of the 75461 peripheral power driver.

There are additional peripheral drivers available that employ different logic functions, such as NAND, OR, and NOR. Furthermore, some peripheral drivers have CMOS compatible inputs. Several popular peripheral driver ICs are listed in Table 5-2. These devices are manufactured by National Semiconductor; however, other manufacturers market similar devices.

Table 5-2: Several commercial peripheral driver devices marketed by National Semiconductor.

Device	Drivers/ Package	Input Compatibility (Logic)	Logic Function	Max. Sink Current	Max. Pull-Up Voltage
DS75450	2	TTL	AND	300	20
DS75451	2	TTL	AND	300	20
DS75452	2	TTL	NAND	300	20
DS75453	2	TTL	OR	300	20
DS75454	2	TTL	NOR	300	20
DS75460	2	TTL	AND	300	30
DS75461	2	TTL	AND	300	30
DS75462	2	TTL	NAND	300	30
DS75463	2	TTL	OR	300	30
DS75464	2	TTL	NOR	300	30
DS3631	2	CMOS	AND	300	40
DS3632	2	CMOS	NAND	300	40
DS3633	2	CMOS	OR	300	40
DS3634	2	CMOS	NOR	300	40
DS3611	2	TTL/CMOS	AND	300	50
DS3612	2	TTL/CMOS	NAND	300	50
DS3613	2	TTL/CMOS	OR	300	50
DS3614	2	TTL/CMOS	NOR	300	50

Transistor Arrays

Unlike a peripheral driver, a transistor array does not have any internal logic functions. A transistor array is simply an IC that contains several integrated driver transistors. The inputs of a transistor array are usually TTL or CMOS compatible, and their outputs are used to control relatively high current and high voltage peripheral devices.

A typical transistor array is shown in Figure 5-41. This device is the CA3096 manufactured by RCA. Notice that it contains five separate driver transistors (three NPN and two PNP). Each transistor is capable of dissipating 200 milliwatts of power. However, the total power dissipation for the device cannot exceed 750 milliwatts.

A big advantage of using transistor arrays over discrete transistors in a control circuit is that the transistors within an array have very close electrical and thermal characteristics. This is because they are integrated into the same chip.

Some transistor arrays include diodes that can be used as suppression diodes for relay and small motor control applications. In addition, different arrays are available to be used with different logic families such as TTL and CMOS. For more application information, you must consult the manufacturers of these devices. Manufacturers of transistor arrays include RCA, National Semiconductor, and Sprague, among others.

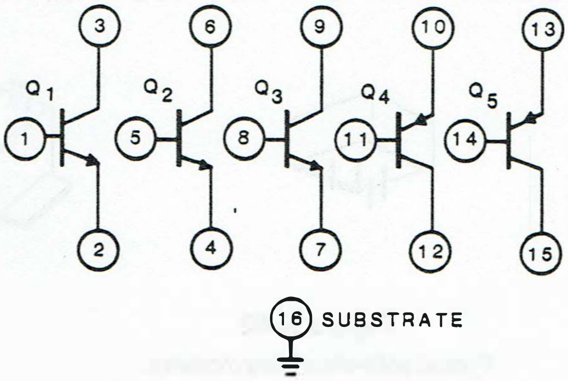


Figure 5-41

A typical transistor array, the RCA CA3096.

Solid-State Relays

Solid-state relays are becoming very popular for on/off control of both DC and AC loads. Several typical solid-state relay modules are pictured in Figure 5-42. Notice their simplicity. There are two input terminals that are TTL compatible for direct control by digital ICs. The load is connected directly across the two output terminals. The input is optically isolated from the output to provide several kilovolts of electrical isolation between the digital and load circuits. Optical isolation will be discussed shortly.

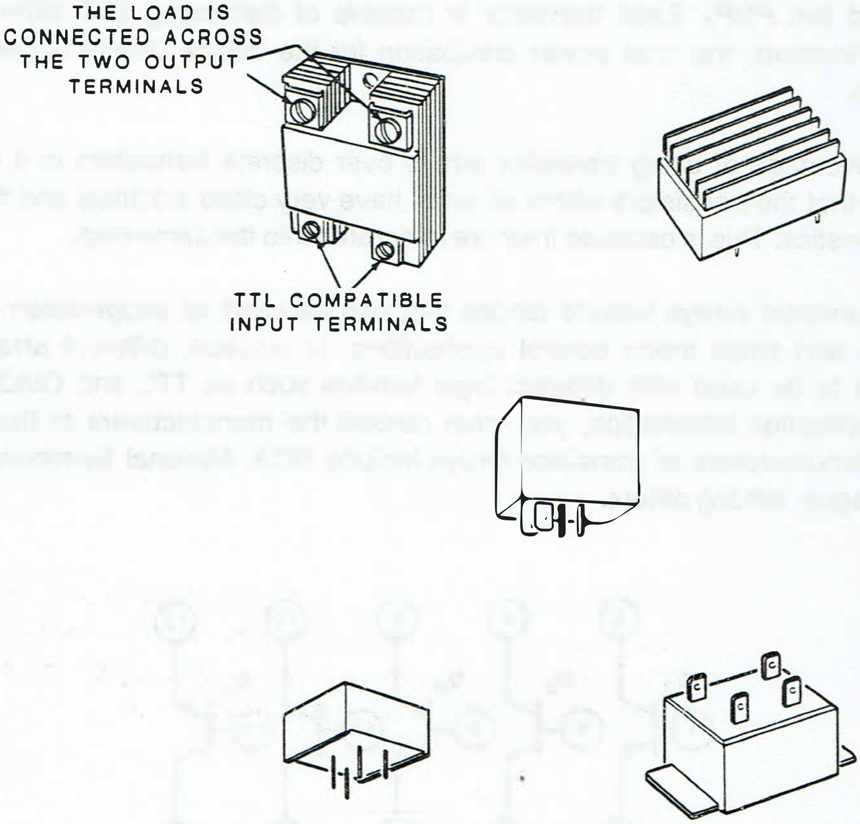

THE LOAD IS CONNECTED ACROSS THE TWO OUTPUT TERMINALS
TTL COMPATIBLE INPUT TERMINALS

Figure 5-42

Typical solid-state relay modules.

Solid-state relays are available to handle load currents up to 50 amperes. However, their cost is in direct proportion to their current carrying capability. The output load voltage rating of a solid-state relay is usually specified as a range of voltages. For example, a given AC relay might specify its output voltage rating as 24-140 volts (rms), while a second relay might specify a 48-280 volts (rms) load voltage rating. Obviously, the second relay could be used to control both 120-volt and 240-volt AC loads, while the first relay is limited to 120-volt loads. A typical load voltage rating for a DC relay is 3-50 volts.

When solid-state relays are used to control heavy AC loads, it is important that the relay not be energized during the peak of the AC cycle. If this happens, the instantaneous high current pulse might exceed the relay current rating and damage the relay. Special devices called **zero-voltage switching** solid-state relays are available for high AC current applications. These devices are used in the same way as standard solid-state relays. However, they include internal circuitry that delays the relay from turning-on until it detects the zero crossing point of the AC voltage.

Another consideration for controlling high DC or AC current levels is heat sinking. The internal components of a solid state relay dissipate a considerable amount of heat when conducting high current levels. If proper heat sinking is not provided, the solid-state relay will not conduct to its rated current level, and it could be damaged. Heat sinking usually involves mounting the relay on an aluminum heat sink block. Consult the manufacturer of the relay for the recommended heat sinking precautions.

A solid-state relay is easily connected to a digital circuit as shown in Figure 5-43. One side of the relay input is grounded and the other side connected directly to a digital circuit output. The relay is normally activated by a logic 1 on its input. When the relay is activated, it completes the DC or AC load circuit.

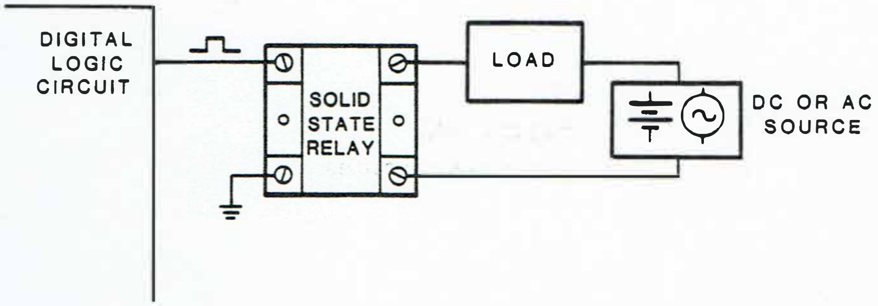


Figure 5-43

A solid-state relay can be connected to any digital output line.

Finally, if the solid-state relay is controlling a reactive load, such as a motor, the relay output should be protected with a **snubber circuit**. A typical RC snubber circuit is shown in Figure 5-44. This circuit dissipates any **back emf** voltage, a reverse voltage spike, generated by the reactive load when the relay is turned off. Without a snubber circuit, a back emf voltage spike could cause damage to the relay or, at a minimum, cause the relay to turn back on. Some solid-state relays have an internal snubber circuit. With these devices, no external components are required.

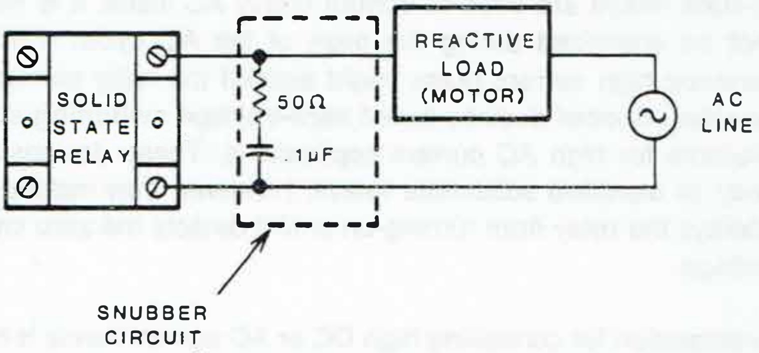


Figure 5-44

A snubber circuit should be used across the output of a solid-state relay when you are controlling active loads such as motors.

Electrical Control Devices, or Actuators

We will define an electrical control device, or actuator, to be any electro-mechanical type of control device. Actuators include electro-mechanical relays, solenoids, and motors.

ELECTRO-MECHANICAL RELAYS

Electro-mechanical relays are available in all shapes and sizes. Some typical relays are pictured in Figure 5-45. They range from small reed relays for low current applications to heavy duty power relays.

Small reed relays are usually sealed in a dual in-line package, or DIP, for mounting in standard IC sockets (Figure 5-45 a and b). Reed relays are generally rated for less than one ampere of load current, with 0.1, 0.25, and 0.5 amps being typical current rating values. Load voltages up to 200 volts DC and 140 volts AC (rms) can be switched by many of these devices.

Input control voltage levels are usually 5, 12, and 24 volts. Consequently, a reed relay can be controlled using an open collector TTL output. Even with a 5 volt reed relay, an open collector device is required to provide sufficient current to energize the relay.

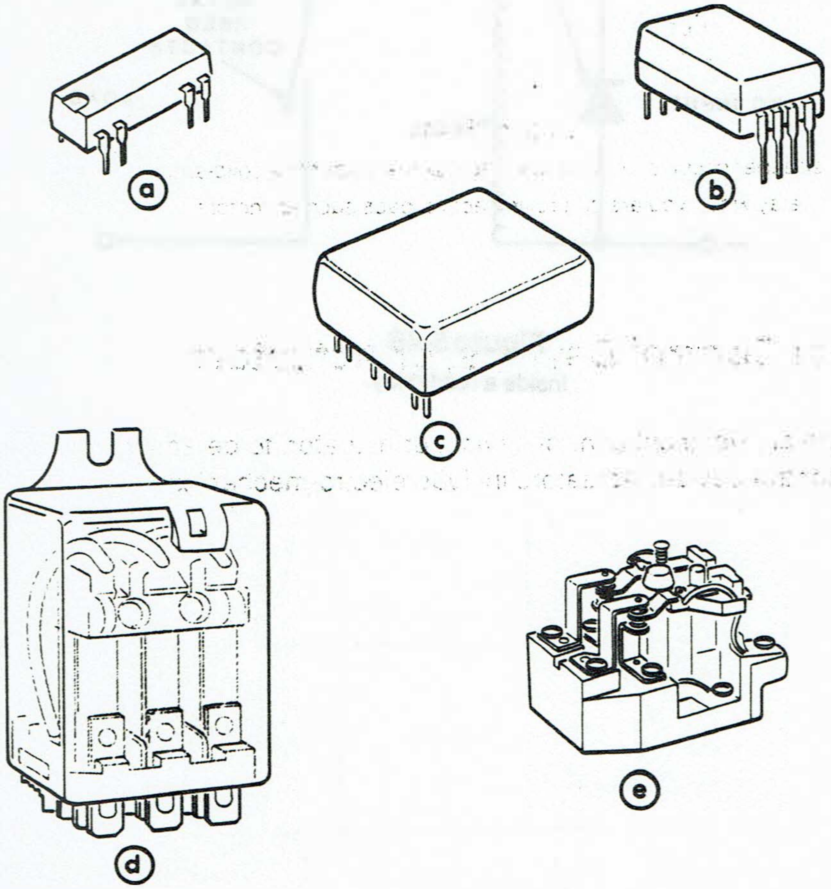


Figure 5-45

Typical electro-mechanical type relays.

The internal structure of a simple reed relay is shown in Figure 5-46. As you can see, the input side consists of a small coil. An optional internal diode in parallel with the coil will suppress any back emf generated when the coil is de-energized. Energizing the coil by application of the required DC input voltage level causes a small set of reed contacts to close (normally open) or to open (normally closed).

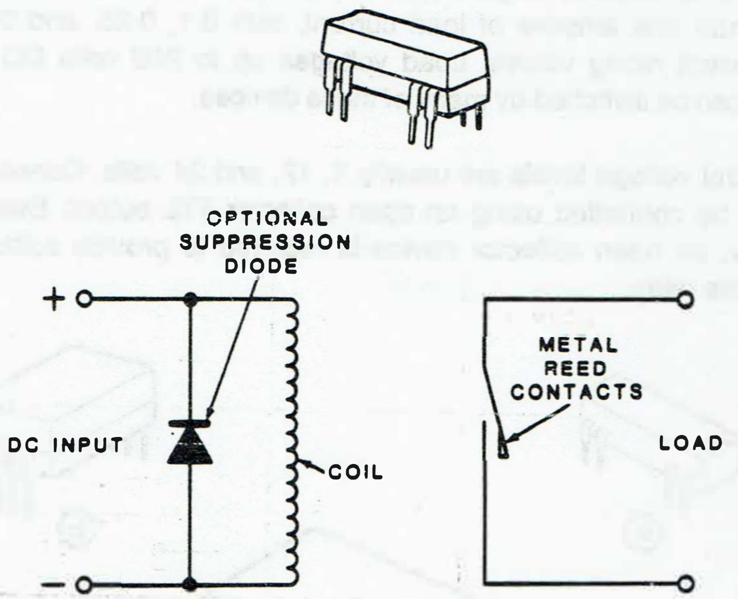


Figure 5-46
Inside a reed relay.

Reed relays are available with several different contact and pin configurations. A few of the more common configurations are illustrated by the contact/pin diagrams in Figure 5-47. Notice that there is a standard code associated with each contact arrangement. For example, 1 Form A (1A) is a normally open single-pole/single-throw, or SPST-NO, relay. On the other hand, a 1 Form B (1B) relay is a normally closed single-pole/single-throw, or SPST-NC, relay. Other options include type 2A, or normally open double-pole/single-throw (DPST-NO), type 1C, or single-pole/double-throw (SPDT), and type 2C, or double-pole/ double-throw (DPDT).

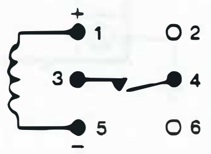
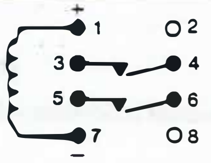
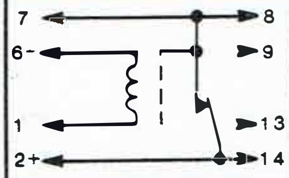
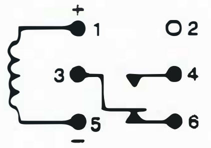
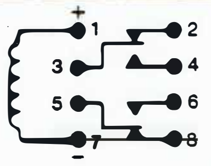
CONTACT ARRANGEMENT		CIRCUIT/PIN DIAGRAM
CODE	DESIGNATION	
1 FORM A	SPST-NO	
2 FORM A	DPST-NO	
1 FORM B	SPST-NC	
1 FORM C	SPDT	
2 FORM C	DPDT	

Figure 5-47

Reed relays are available with several different contact and pin configurations.

A high power relay must be used to control load currents above one ampere. High power electro-mechanical relays are available in a variety of power ratings and contact configurations to control both DC and AC current levels up to 50 amperes. Some power relays require DC excitation, while others require AC excitation. Since power relays cannot be controlled directly by digital signals, an intermediate solid-state relay must be used as shown in Figure 5-48. Notice that the solid-state relay is activated by the digital output. The intermediate solid-state relay then energizes the high power relay.

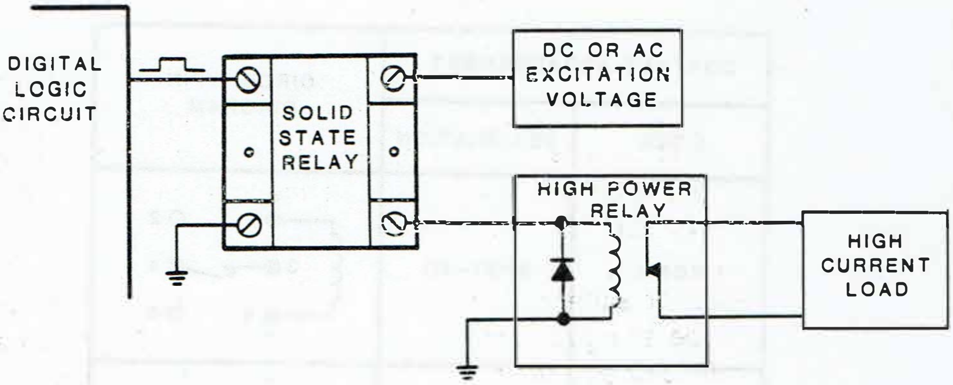


Figure 5-48
An intermediate solid-state relay is used to energize a high power relay.

SOLENOIDS

A solenoid, pictured in Figure 5-49, is an electro-mechanical actuator that converts electrical energy into a linear mechanical motion. Solenoids are used in applications where a sudden force must be applied to an object. A solenoid consists of a plunger surrounded by a coil. The plunger is made of a ferrous material, such as hard steel, and may be free or spring loaded. Application of an excitation current to the solenoid coil creates a magnetic field, which exerts a force on the plunger and causes it to move, or actuate.

There are both DC and AC solenoids. A DC solenoid requires a DC excitation current, while an AC solenoid requires an AC excitation current. Like heavy duty relays, solenoids must be energized using an intermediate solid-state relay for digital control. The digital circuit activates the intermediate solid-state relay which, in turn, passes the excitation current to the solenoid.

Solenoids are rated for either intermittent or continuous duty. Intermittent duty solenoids can only be energized for a given amount of time. On the other hand, continuous duty solenoids can be energized continuously without damage.

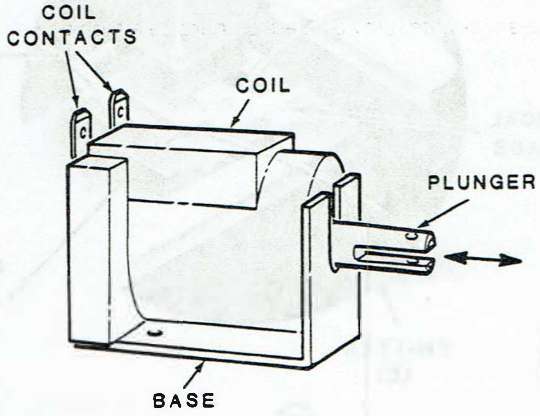


Figure 5-49
A solenoid is an electro-mechanical actuator that converts electrical energy into a linear mechanical motion.

Optocouplers

Optocouplers, or **optoisolators**, can be classified as control devices, since they are used by a digital system to control external events. They are extremely important in protecting digital circuits and their human operators from high power load circuits. Most optoisolators can provide several kilovolts of isolation between the digital control and load circuits. This means that it would take several kilowatts to break down the optical isolation and damage the digital control circuit.

Optoisolators operate using an infrared LED, called an **IRED**, and a phototransistor as shown in Figure 5-50a. Compare this to the schematic representation shown in Figure 5-50b. Notice that the IRED and phototransistor are separated by a layer of optical-quality glass and that they are coupled only by the infrared light beam produced by the IRED. The amount of electrical current applied to the IRED terminals determines the intensity of the IRED. This, in turn, controls the conductivity of the phototransistor.

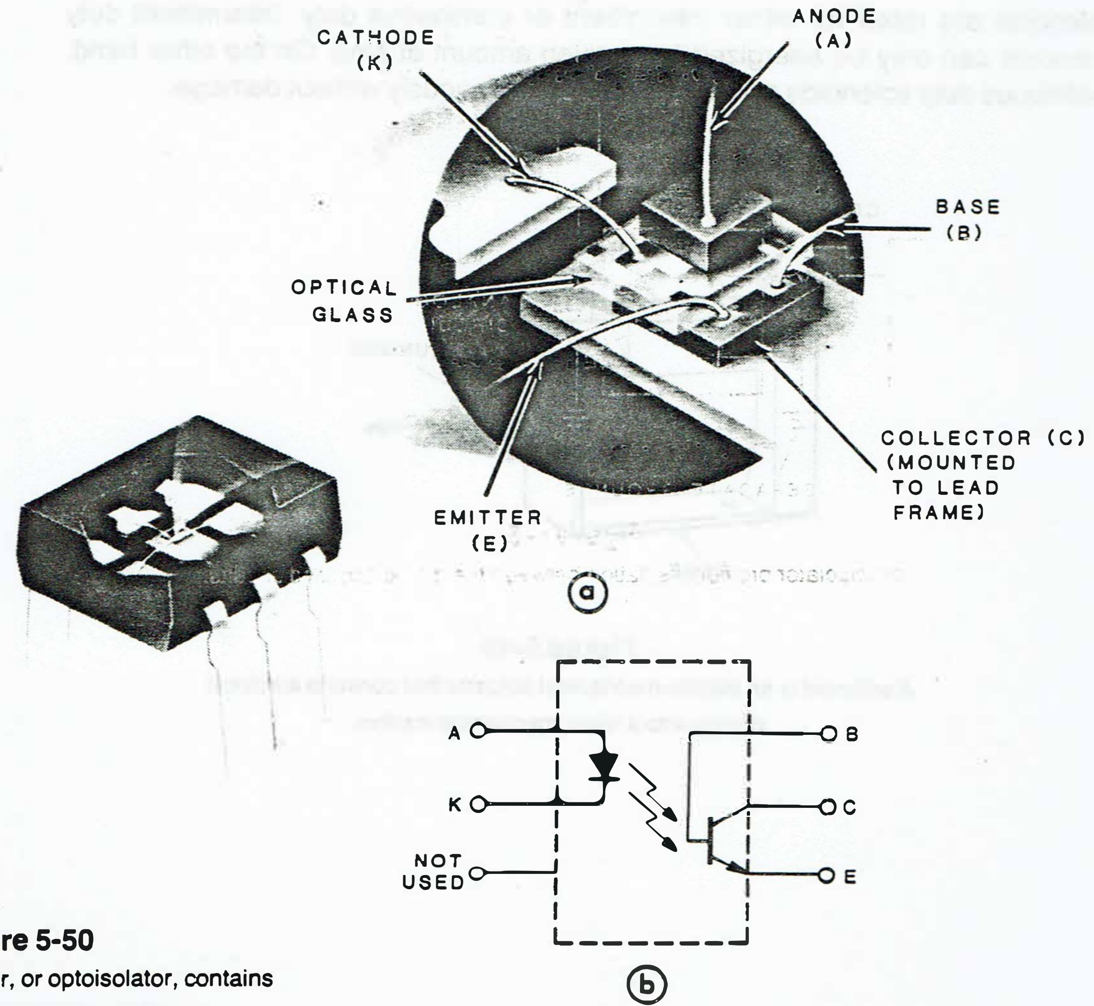


Figure 5-50
A typical optical coupler, or optoisolator, contains an IRED and a phototransistor.

Since the input diode and output transistor are completely separated, they need not use the same power source or ground return paths. This makes it possible to greatly reduce or eliminate electromagnetic interference, leakage currents, ground loop currents, and other undesirable features associated with standard electrical or electromagnetic coupling.

The most important feature of an optoisolator is the electrical isolation that it provides between a digital control circuit and a higher power analog load circuit. As you can see in Figure 5-51, the optoisolator **is the** interface between the digital and load circuits. Here, a logic 1 generated by the digital circuit turns-on the IRED, which causes the phototransistor to conduct and supply current to the load. Conversely, a logic 0 turns-off the IRED, the phototransistor acts like an open switch and no current is passed through the load. Thus, the optoisolator is acting like a digitally controlled switch for the load.

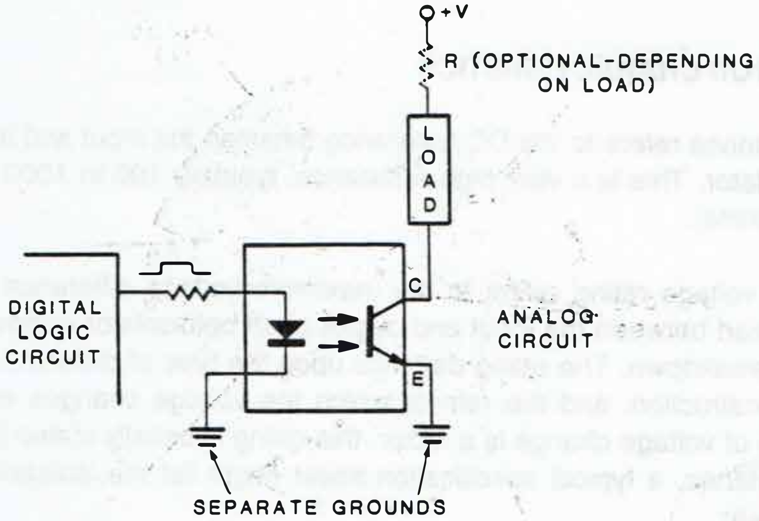


Figure 5-51

An optoisolator provides isolation between the digital and load circuits.

Observe that only the collector (C) and emitter (E) leads of the phototransistor are used. The base (B) lead is not required and is often "clipped-off" for this application. In addition, the input and output lines of the optoisolator can operate from two separate grounds. Isolated grounds will prevent lethal shock-hazard situations. Furthermore, the digital circuit and its operator are protected against the higher voltages of the load circuit, since there is no direct connection between the two circuits.

Optical isolation is especially advantageous in biomedical equipment, where optocouplers are used to provide electrical isolation between the patient and potentially dangerous voltages within the instrument. Also, since the diode and the silicon transistor cannot reverse their roles, the light path through the optocoupler is strictly one way. This makes it possible to control industrial machine circuits with computer logic circuits without switching transients and power surges being fed back into the delicate computer devices.

OPTOISOLATOR CHARACTERISTICS

Isolation resistance refers to the DC resistance between the input and the output of the optoisolator. This is a very high resistance, typically 100 to 1000 gigohms (10^{11} to 10^{12} ohms).

The isolation voltage rating refers to the maximum voltage difference that can be safely applied between the input and output of an optoisolator without danger of insulation breakdown. The rating depends upon the type of dielectric material, method of construction, and the rate at which the voltage changes over time. Since the rate of voltage change is a factor, this rating is usually stated in several ways. For instance, a typical specification sheet might list the isolation voltage rating as follows:

Surge Isolation Voltage (Input to Output)
1500 volts (peak), 1060 volts (rms)

Steady-State Isolation Voltage (Input to Output)
950 volts (peak), 660 volts (rms)

I should mention that these devices are not self-healing. If the isolation voltage rating is exceeded to the point that insulation breakdown occurs, the device is permanently damaged. At the very least, a carbonized resistive path is formed as a result of high currents passing through the insulation material. In extreme cases, lead wires will melt, resulting in a possible short circuit between input and output connections. Therefore, when you use an optocoupler to isolate computer logic circuits, you should select an optocoupler with a sufficiently high voltage rating. Under extreme conditions, additional fuse protection may be necessary.

SOME TYPICAL OPTOISOLATOR DEVICES

Now that you are familiar with the major operational characteristics of optoisolators, let's take a quick look at some of the different types that are available.

One of the earliest and still most widely used general-purpose optoisolator is the IRED input/phototransistor output version that you just learned about. This device is a good compromise between speed and output current.

The **photo-Darlington** optoisolator, shown in Figure 5-52, is available where more output current is needed at a relatively low switching speed (30 kHz or less). This device uses a Darlington transistor arrangement on the output side of the optoisolator. A Darlington transistor circuit has a higher current gain than a single transistor circuit. As a result, the photo-Darlington will conduct about ten times more current than that of a single phototransistor.

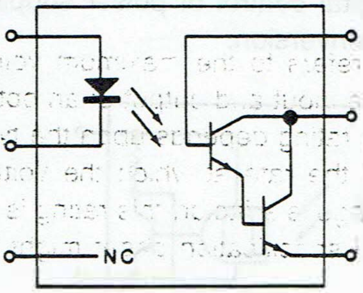


Figure 5-52
A photo-Darlington optoisolator can conduct more output current than a single phototransistor output.

A relatively new device is the **FET optoisolator**, shown in Figure 5-53. This device has a MOS transistor for its output. It is used primarily to control low level AC and DC analog signals where minimal power drain is required. However, with switching times less than 15 microseconds, the FET optoisolator is also useful in high speed analog switch applications.

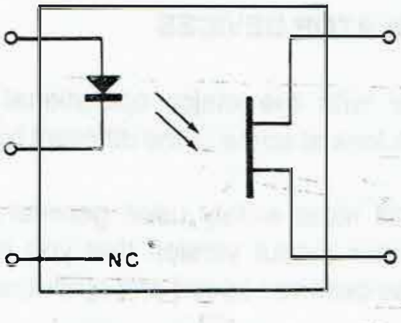


Figure 5-53

An FET optoisolator.

The **digital logic optoisolator**, shown in Figure 5-54, is representative of a family of optocouplers available for use in digital control applications. The Schmitt-trigger output provides a fast switching action that is ideal for interfacing computers to peripheral equipment, digital control of power supplies, motors, and servos, as well as simple logic level conversion.

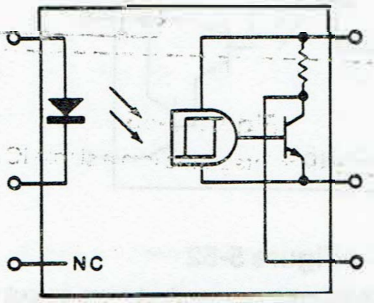


Figure 5-54

A digital logic optoisolator.

As you can see from the preceding discussion, a wide-range of optoisolators are available to perform many tasks. Other special-purpose devices are available, as well as many variations of these common types. One example is the placement of several coupling devices within a single package as shown in Figure 5-55. This device contains four separate high-speed, high-gain optocouplers in one 16-pin DIP package.

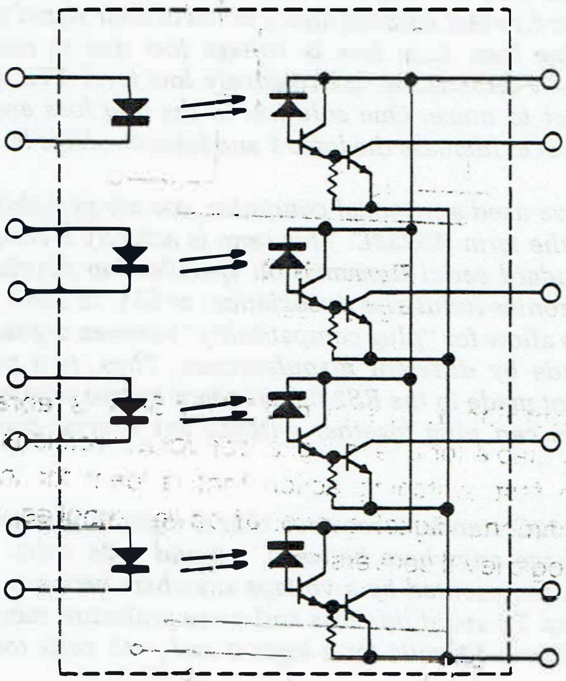


Figure 5-55

Multiple optoisolators integrated into a single IC package.

Example 5-5

When learning about multiplexers, you were introduced to serial data transmission. Recall that logic 1's and 0's were transmitted and received one at a time. The purpose of serial transmission is to save conductor paths when transmitting digital data over long distances.

As it turns out, TTL signals cannot be used for transmission much over five feet without losing much of their signal strength due to ***line loss****. Line loss is voltage loss due to conductor resistance. Furthermore, the relatively low level TTL voltages are subject to noise. One solution to the line loss and noise problems is to increase the logic 1 and logic 0 voltage levels.*

If you have used a personal computer, you are probably familiar with the term ***RS232C****. This term is actually a designation for a standard serial transmission specification developed by the Electronics Industries Association, or EIA, in 1969. Its purpose is to allow for "plug compatibility" between digital equipment made by different manufacturers. Thus, two pieces of equipment made to the RS232C standard by two separate manufacturers can plug together without any special conversion circuitry.*

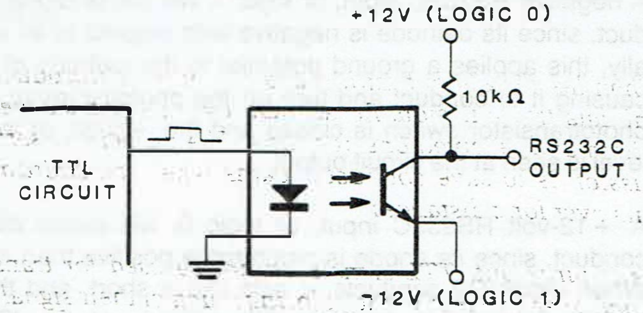
The RS232C standard requires that a logic 0 be represented by a voltage anywhere between +3 and +25 volts. A logic 1 is then represented by a voltage anywhere between −3 and −25 volts. To avoid line loss and noise problems, most digital circuits use +12 volts for a logic 0 and −12 volts for a logic 1.

Now, the problem is this: a TTL circuit uses 0 volts and +5 volts to represent logic 0's and 1's, while RS232C requires +12 volts and −12 volts to represent the same respective logic. Consequently, a converter circuit must be designed to convert between the two different logic levels.

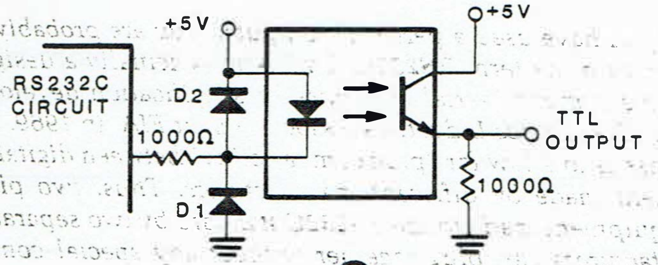
Design a converter circuit using optoisolators to convert between TTL and RS232C, and between RS232C and TTL.

First, we will design a circuit to convert from TTL to RS232C. The optoisolator is ideal for this circuit, since it will optically separate the two voltage levels.

We will use the TTL levels on the input side of the optoisolator to turn the IRED on and off, as shown in Figure 5-56a. A logic 1 will turn the IRED on, while a logic 0 will turn it off.



ⓐ
TTL-TO-RS232C CONVERTER



ⓑ
RS232C-TO-TTL CONVERTER

Figure 5-56
Optoisolators are used as level converters for serial data transmission (Example 5-5).

The RS232C levels will be connected to the collector (C) and emitter (E) of the output phototransistor as shown. Notice that the +12-volt logic 0 potential is connected to the collector via a pull-up resistor and the −12-volt potential is connected directly to the emitter.

Now, as long as a logic 0, or ground, potential is applied to the optoisolator input, the IRED does not illuminate and the phototransistor does not conduct. Thus, the phototransistor acts like an open switch in a pull-up resistor circuit and a +12-volt, or logic 0, RS232C level is seen at the output.

When a logic 1 TTL level, or +5 volts, is applied, the phototransistor conducts and acts like a closed switch. As a result, the −12-volt, or logic 1, RS232C level is seen at the bottom of the pull-up resistor.

The RS232C-to-TTL converter in Figure 5-56b uses a diode arrangement on the input and a pull-down resistor on the output. Here's how it works.

A negative RS232C input, or logic 1, will cause diode D1 to conduct, since its cathode is negative with respect to its anode. Ideally, this applies a ground potential to the cathode of the IRED, causing it to conduct and turn on the phototransistor. Thus, the phototransistor switch is closed and the +5volt, or logic 1, TTL level is seen at the circuit output.

A +12-volt RS232C input, or logic 0, will cause diode D2 to conduct, since its anode is much more positive than its cathode. When diode D2 conducts, it acts like a short, and the +5-volt supply potential is seen on the cathode of the IRED. Consequently, the IRED will not conduct, causing the phototransistor to act like an open switch. This results in a ground, or logic 0, at the TTL output.

Self-Test Review

21. What is a peripheral power driver? ____________________

22. What is a transistor array? ____________________

23. Explain why a transistor array might be more desirable than using single, or discrete, drive transistors. ____________________

24. List at least three considerations you must take into account when you are using solid-state relays to control high current loads.

A. ____________________

B. ____________________

C. ____________________

25. Describe the physical and electrical characteristics of a typical reed relay.

26. The greatest advantage of an optoisolator is the ____________ ____________ provided between its input and output.

27. Describe the construction of a general purpose optoisolator. ____________

28. Design an optoisolator circuit that will activate a load when a logic 0 is applied to the optoisolator.

29. Another way to transmit serial data is by using current. The presence of a given current level indicates a logic 1, while the absence of any current represents a logic 0.

In Example 5-5, you learned about the RS232C voltage standard. It turns out that RS232C voltage levels are only used for serial data communications up to 50 feet, due to excessive line loss beyond this distance. For distances over 50 feet and up to 1 mile, a current standard must be used. The most common current standard is the 20 milliamp standard. A logic 1 is represented by the presence of a 20 milliamp current, while a logic 0 is represented by the absence of current.

Since current will only flow in a closed loop, a switch placed in series with a 20 milliamp current loop can be used to make and break the loop, thereby generating logic 1's and 0's. This idea is illustrated in Figure 5-57, parts a and b.

Recall that an optoisolator is nothing more than a controlled switch. Using this idea, design an optoisolator circuit that will convert between TTL and the 20 milliamp current loop standard.

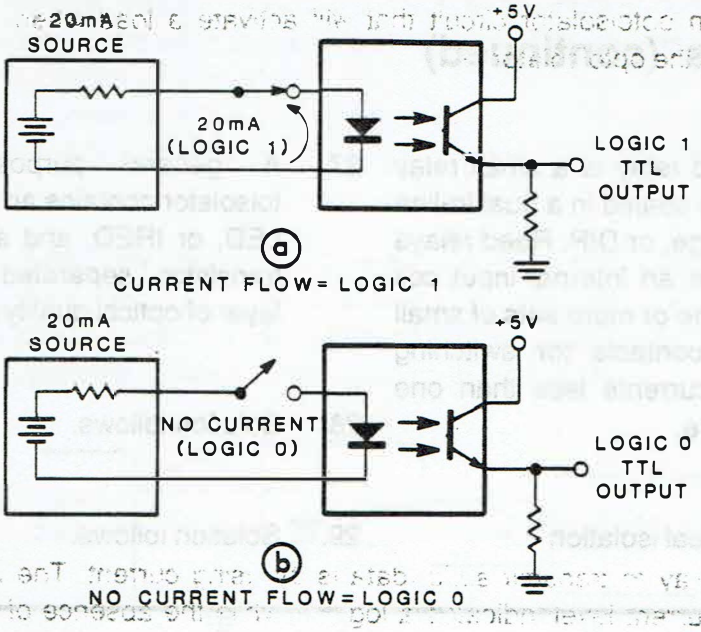


Figure 5-57

A switch in series with a 20 mA current loop can be used to generate logic 1's and 0's.

Answers

21. It is a device that includes both logic gates and drive transistors within the same IC package.

22. It is an IC that contains several integrated transistors.

23. Because the integrated transistors in an array have very close electrical and thermal characteristics.

24A. Zero-voltage switching.

24B. Heat sinking.

24C. Protecting the relay from any back emf generated by a reactive load such as a meter.

Answers (continued)

25. A reed relay is a small relay device sealed in a dual in-line package, or DIP. Reed relays contain an internal input coil and one or more sets of small reed contacts for switching load currents less than one ampere.

26. electrical isolation

27. A general purpose optoisolator contains an infrared LED, or IRED, and a phototransistor separated by a layer of optical quality glass.

28. Solution follows.

29. Solution follows.

Solution to Questions 28 and 29.

28. Refer to Figure 5-58. Notice that a logic 0 applied to the optoisolator causes the IRED to conduct and activate the load via the phototransistor. With a logic 1, the IRED is not forward-biased and does not conduct. As a result, the phototransistor is turned off and no current will flow in the load circuit.

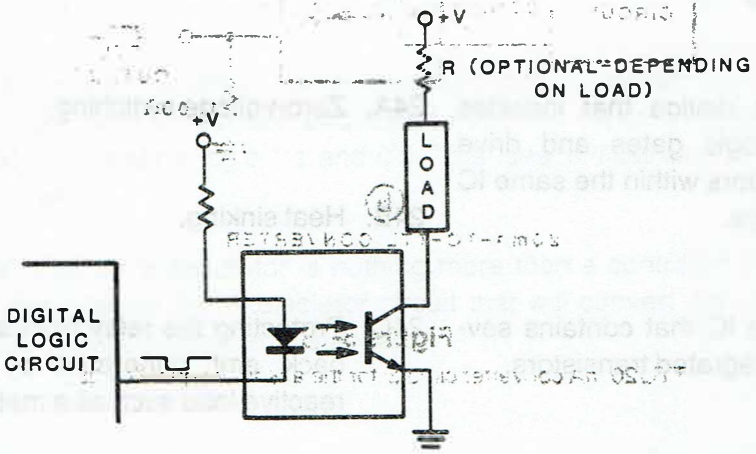
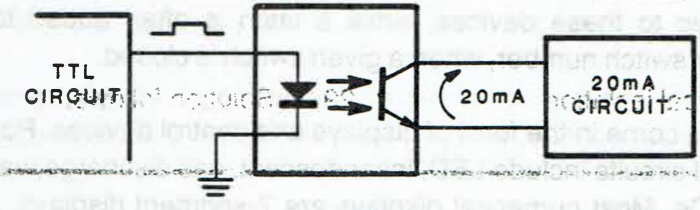


Figure 5-58

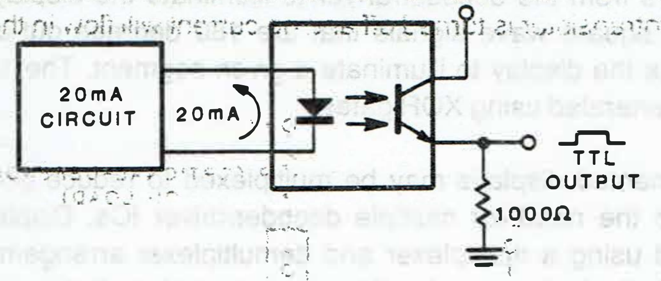
The circuit design for the answer to question 28.

29. In Figure 5-59a, a logic 1 TTL signal causes the IRED to conduct, emitting infrared light to the base of the phototransistor. This causes the phototransistor to conduct and close the 20 mA current loop, creating a logic 1 in the loop. When a logic 0 TTL signal is present, the optoisolator breaks the current loop, resulting in no current flow which represents a logic 0 in the loop.

In Figure 5-59b, the presence of 20 mA turns on the optoisolator and creates a +5-volt, or logic 1, TTL level at the top of the pull-down resistor. The absence of any current in the loop turns off the optoisolator, and a ground, or logic 0, potential is seen at the top of the pull-down resistor.



ⓐ
TTL-TO-20mA CONVERTER



ⓑ
20mA-TO-TTL CONVERTER

Figure 5-59

TTL/20 mA converter circuits for the answer to question 29.

SUMMARY

In this unit, you learned how to input binary information to a digital circuit via single switches or switch matrices, called keyboards. Switches are connected in a pull-up or pull-down resistor circuit to generate binary data. Mechanical switches must be debounced using debouncing circuits to eliminate a succession of 1's and 0's caused by the vibrating mechanical switch contacts.

Keyboards are nothing more than a matrix of switches. A keyboard decoding circuit must be designed to detect when a switch is closed and determine exactly which switch in the matrix is closed. Keyboard decoding circuits scan the switch matrix using a decoder and multiplexer arrangement. A counter is used to apply the select logic to these devices, while a latch is often added to capture the count value, or switch number, when a given switch is closed.

Output devices come in the form of displays and control devices. Popular displays used for digital circuits include LED, incandescent, gas discharge, vacuum fluorescent, and LCDs. Most numerical displays are 7-segment displays. Alphanumeric displays are usually dot matrix displays.

A device called a decoder/driver is used to convert BCD data into the display code required to illuminate 7-segment displays. All displays, except LCDs, require logic 1's or 0's from the decoder/driver to illuminate the display segments. With an LCD, two square wave signals that are 180 degrees out-of-phase must be applied across the display to illuminate a given segment. The 180 degree phase shift can be generated using XOR gates.

Several segmented displays may be multiplexed to reduce power consumption and eliminate the need for multiple decoder/driver ICs. Display multiplexing is accomplished using a multiplexer and demultiplexer arrangement that turns on the individual displays, one at a time, in succession. If the multiplexing rate is at least 30 Hz, the display looks constant due to the image retention of your eye.

Dot matrix displays are used to display alphanumeric information. The dot matrix consists of rows and columns connected by discrete LEDs or incandescent lamps. The rows and columns are scanned using a decoder and character generator ROM. The decoder selects the dot matrix row, while the ROM generates the data required to illuminate the LEDs or lamps in a selected row.

Digital control devices include driver transistors, peripheral power drivers, transistor arrays, solid-state relays, electro-mechanical relays, and optoisolators. Most of these devices are used to control high power AC and DC loads from the relatively low power output signals of a digital circuit. Optoisolators are especially important because they can be used to isolate the digital circuit and its operator from hazardous high power load voltages and currents.

UNIT EXAMINATION

1. A pushbutton switch, like those used in digital keyboards, acts like a:
   A. STSP switch.
   B. DPST switch.
   C. SPDT switch.
   D. SPST switch.

2. To debounce a SPDT switch you can use:
   A. A D flip-flop.
   B. Cross-coupled NOR gates.
   C. Cross-coupled inverters.
   D. All of the above.

3. When using a Schmitt-trigger circuit to debounce a pushbutton switch with a 2 millisecond bounce period, the RC time constant should be at least:
   A. 0.2 mS.
   B. 2 mS.
   C. 20 mS.
   D. 200 mS.

4. If a 0.1 µF capacitor is used in the Schmitt-trigger debouncing circuit in question 3, the resistor value must be at least:
   A. 0.2 MΩ.
   B. 200 Ω.
   C. 20 kΩ.
   D. None of the above.

5. When switch 15 is depressed in Figure 5-60, the 4-bit latch is loaded with the value:
   - A. 0000.
   - B. 1001.
   - C. 1111.
   - D. 1110.

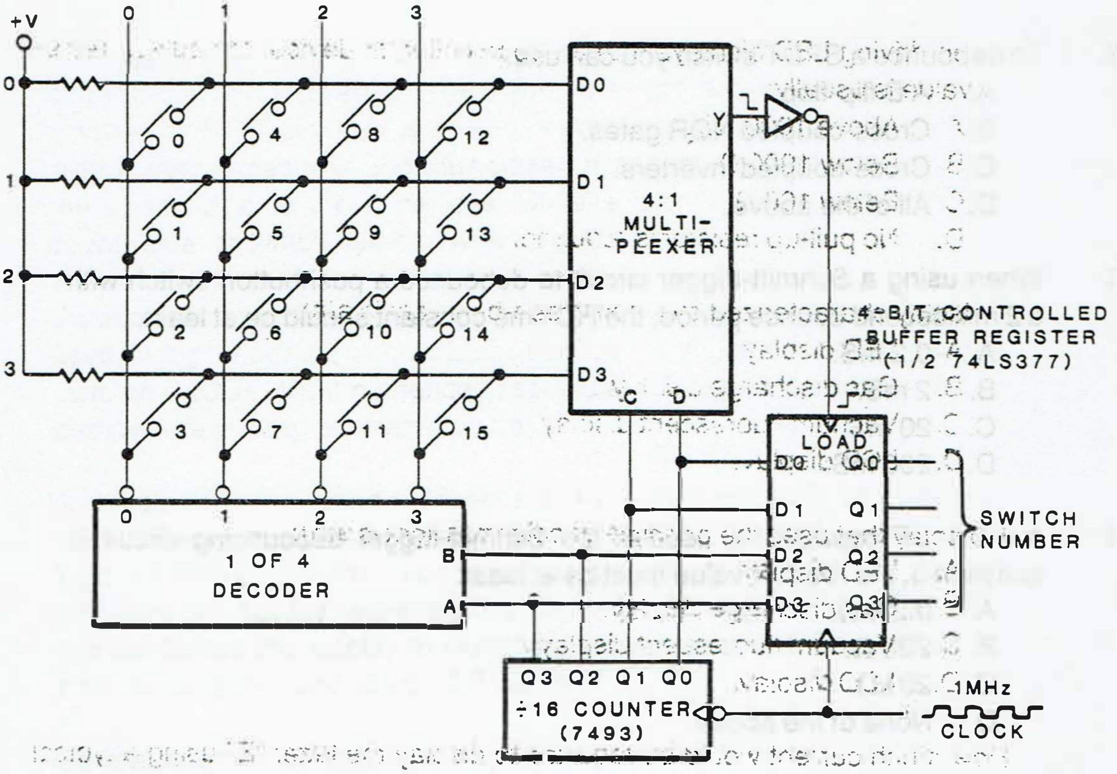


Figure 5-60

A keyboard decoding circuit for Exam question 5.

6. An incandescent lamp is rated at 12 volts and 40 milliamps. Which of the following TTL devices will provide optimum brightness when driving this lamp?
    A. A 7407 hex buffer.
    B. A 7405 hex inverter.
    C. A 7404 hex inverter.
    D. A 7400 quad 2-bit NAND.

7. When driving LED displays with an open collector device, the pull-up resistor value is usually:
    A. Above 1000 Ω.
    B. Below 1000 Ω.
    C. Below 100 Ω.
    D. No pull-up resistor is required.

8. A display characterized by a soft green-blue light is a(n):
    A. LED display.
    B. Gas discharge display.
    C. Vacuum fluorescent display.
    D. LCD display.

9. A display that uses the principle of polarized light is a(n):
    A. LED display.
    B. Gas discharge display.
    C. Vacuum fluorescent display.
    D. LCD display.

10. How much current would be required to display the letter "E" using a typical 7-segment LED display?
    A. 20 mA.
    B. 80 mA.
    C. 100 mA.
    D. 200 mA.

11. The ripple blanking feature of a decoder/driver:
    A. Inserts a decimal point in the display.
    B. Eliminates trailing zeros.
    C. Eliminates leading zeros.
    D. Eliminates intermediate zeros.

12. A common CMOS 7-segment display decoder/driver is the:
    - A. CD4511.
    - B. 74LS47.
    - C. 74LS48.
    - D. CD4543.

13. Any number of displays can be driven with a single decoder/driver using:
    - A. Full encoding.
    - B. Display multiplexing.
    - C. Display decoding.
    - D. A separate decoder/driver must be used for each individual 7-segment display.

14. To illuminate an LCD segment:
    - A. Around 20 mA of current must be passed through the segment.
    - B. Neon gas must be ionized.
    - C. Phosphors must be bombarded with electrons.
    - D. The segment signal must be 180° out-of-phase with a reference backplane signal.

15. The gate used to drive the segments of an LCD display is:
    - A. Any open collector gate.
    - B. An XOR gate.
    - C. A NAND gate.
    - D. Driver transistors must be used in place of a gate output.

16. The control device used for a dot matrix display is:
    - A. A peripheral power driver.
    - B. A ROM.
    - C. A decoder/driver.
    - D. A transistor array.

17. A control device that includes both logic gates and driver transistors is the:
    - A. Peripheral power driver.
    - B. Transistor array.
    - C. Solid-state relay.
    - D. None of these.

18. A control device that employs an infrared LED and a photo-transistor is the:
    - A. Reed relay.
    - B. Peripheral power driver.
    - C. Transistor array.
    - D. Optoisolator.

19. The optoisolator that can be used to conduct higher levels of current is the:
    - A. IRED input/phototransistor output optoisolator.
    - B. Photo-Darlington optoisolator.
    - C. FET optoisolator.
    - D. Digital logic optoisolator.

20. Which of the following logic representations might you find in an RS232C system?
    - A. Logic 0 = 0V, logic 1 = +5 V.
    - B. Logic 0 = 0 mA, logic 1 = 20 mA.
    - C. Logic 0 = +12 V, logic 1 = −12 V.
    - D. Logic 0 = −12 V, logic 1 = +12 V.

EXAMINATION ANSWERS

1. D — A pushbutton switch, like those used in digital keyboards, acts like a SPST switch.

2. D — To debounce a SPDT switch you can use a D flip-flop, cross-coupled NOR gates, or cross-coupled inverters.

3. C — When using a Schmitt-trigger circuit to debounce a push-button switch with a 2 mS bounce period, the RC time constant should be at least 20 mS.

4. A — If a 0.1 μF capacitor is used in the Schmitt-trigger debouncing circuit in question 3, the resistor value must be at least 0.2 MΩ.

5. C — When switch 15 is depressed in Figure 5-60, the 4-bit latch is loaded with the value 1111.

6. A — If an incandescent lamp is rated at 12 volts and 40 milliamps, a 7407 hex buffer will provide optimum brightness when driving this lamp.

7. B — When driving LED displays with an open collector device, the pull-up resistor value is usually below 1000 Ω.

8. C — A display characterized by a soft green-blue light is the vacuum fluorescent display.

9. D — A display that uses the principle of polarized light is the LCD display.

10. C — A typical 7-segment LED requires 20 mA of current to light one segment. Since the letter "E" uses five segments, the LED requires 5 × 20 mA or 100 mA of current to display the letter "E."

11. C — The ripple blanking feature of a decoder/driver eliminates leading zeros.

12. A — A common CMOS 7-segment decoder/driver is the CD4511.

13. B — Any number of displays can be driven with a single decoder/driver using display multiplexing.

14. D — To illuminate an LCD segment, the segment signal must be 180° out-of-phase with the reference backplane signal.

15. B — The gate used to drive the segments of an LCD display is an XOR gate.

16. B — The control device used for a dot matrix display is a ROM.

17. A — A control device that includes both logic gates and driver transistors in a single IC is the peripheral power driver.

18. D — A control device that employs an infrared LED and a phototransistor is the optoisolator.

19. B — The optoisolator that can be used to conduct higher levels of current is the photo-Darlington optoisolator.

20. C — In an RS232C system you would likely find that a logic 0 = +12 V and a logic 1 = −12 V.

UNIT 6

ANALOG I/O, POWER SUPPLIES, AND DIGITAL CIRCUIT PROCESSING

CONTENTS

Introduction . . . . Page 6-3

Unit Objectives . . . . Page 6-4

Unit Activity Guide . . . . Page 6-5

The Analog Connection . . . . Page 6-6

The Power Supply . . . . Page 6-31

Digital Circuit Processing . . . . Page 6-46

Summary . . . . Page 6-86

Unit Examination . . . . Page 6-89

Examination Answers . . . . Page 6-93

INTRODUCTION

Your knowledge of digital circuit design has now progressed from decision making combinational logic and sequential memory circuits to input/output circuits. In the last unit, the two I/O circuits discussed were switches and displays. Both are basically digital in nature. That is, switches are used to input digital information into a digital circuit, and displays are used to output digital information from a digital circuit.

Since most of the real world is an analog world, it is often required to convert between analog and digital signals. For this reason, you must be familiar with devices that allow you to connect, or interface, your digital circuit with the real analog world. These devices are called **digital-to-analog** and **analog-to-digital** converters and are the beginning topic of this unit.

A major part of any digital design is the circuit that must supply power to operate the circuit. This support circuit, called a **power supply**, may be in the form of a simple battery for CMOS circuits or a special module that converts household power into the voltage and current levels required by TTL circuits. In any case, nothing works without the proper power supply. In the second section of this unit, you will learn how to specify and select power supply modules.

Finally, after the circuit has been specified, the final task is to prototype the circuit and make it ready for volume production. To accomplish this you must be acquainted with both hardware and software prototyping techniques. These topics are covered in the last section of this unit.

UNIT OBJECTIVES

When you complete this unit, you should be able to:

1. Describe how analog signals are produced by the binary weighting of a digital word.
2. Define and calculate the step size and % resolution for any DAC.
3. Define DAC accuracy and determine how close a given DAC output will be to its expected output.
4. Describe the difference between a unipolar and bipolar DAC output.
5. Name the two general classes of analog-to-digital converters.
6. Describe the operation of both a parallel output ADC and a serial output V/F converter.
7. List three types of power supplies used for digital circuits.
8. Name the major functional parts of a linear power supply.
9. State the major difference between a linear and switching supply.
10. Calculate the power requirements of a digital circuit.
11. Specify a power supply module to satisfy the power requirements of a digital circuit.
12. List at least three ways to construct a prototype circuit.
13. Explain how to construct wire wrap circuits.
14. Describe several different methods that are used to construct PC boards.
15. List several inspection and quality control considerations for PC boards.
16. Explain how CAD systems are used to design, simulate, and implement digital circuits.
17. Explain when and how to implement special shielding and grounding for digital circuits.

UNIT ACTIVITY GUIDE

☐ Read section on “The Analog Connection.”

☐ Answer Self-Test Review Questions 1-10.

☐ Perform Experiments 15 and 16 in Unit 8.

☐ Read section on “The Power Supply.”

☐ Answer Self-Test Review Questions 11-20.

☐ Read section on “Digital Circuit Processing.”

☐ Answer Self-Test Review Questions 21-30.

☐ Study Summary.

☐ Complete Unit Examination.

☐ Check Examination Answers.

THE ANALOG CONNECTION

Up to this point, you have been working in more or less an artificial digital world. Two state digital circuits are good for making decisions and computations, but often cannot be used to directly sense and control real world events. Think about the real world around you. The world as you know it is an analog world. As the sun rises in the morning the daylight gradually grows brighter and the temperature gradually increases. As the sun begins to set in the evening, the daylight gradually grows dimmer and the temperature decreases. The sunlight does not appear at an instant in the morning, then go out suddenly in the evening. The sunlight, darkness, and temperature are all gradual and not abrupt processes. If the sunlight and temperature were represented by a voltage or current, the voltage or current signal would be continuous, gradually increasing to some maximum level; then gradually decreasing to some minimum level. As you know, such a signal is called an **analog signal**.

You are aware that digital circuits do not operate on this principle. The digital world is a **two-state** world made up of 1's and 0's. Therefore, in order for a digital circuit to communicate with the "real" analog world, a digital-to-analog or analog-to-digital conversion must take place.

Before using a digital circuit to sense and control real world events and processes, you must understand the principles of digital-to-analog and analog-to-digital conversions. A device called a **DAC** is used to provide digital-to-analog conversions. A device called an **ADC** is used to provide analog-to-digital conversions.

Digital-to-Analog Conversion

The digital-to-analog converter, or DAC, is a device that converts the two-state output of a digital circuit to a continuous analog voltage or current signal. The analog output signal provided by the DAC is always proportional to the digital input value. For now, we will simply think of the DAC as a black box, since you do not necessarily have to understand the internal operating principles to effectively apply and use a DAC.

A DAC with four digital input lines and one analog output line is shown in Figure 6-1. A DAC is usually rated by the number of its digital input lines. Each input line provides one bit within the digital value to be converted. Thus, the DAC shown in Figure 6-1 is called a 4-bit DAC.

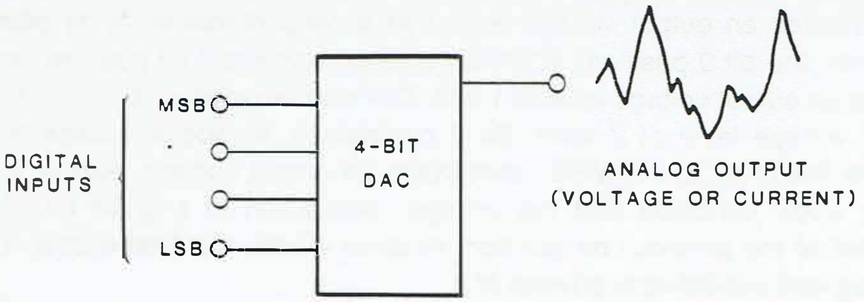


Figure 6-1
A DAC converts a binary input value to a proportional analog voltage or current output value.

For example purposes, let's suppose that the 4-bit DAC in Figure 6-1 has a 0-15 volt output range. Furthermore, let's connect a 4-bit binary up-counter to the digital input lines of the DAC as shown in Figure 6-2. If you allow the counter to count up, each count from 0000 to 1111 will provide a corresponding output voltage level. Ideally, a count of 0000 provides an output voltage of 0 volts; a count of 0001 provides an output voltage of 1 volt; a count of 0010 provides a voltage of 2 volts; and so on, until a count of 1111 produces an output voltage of 15 volts.

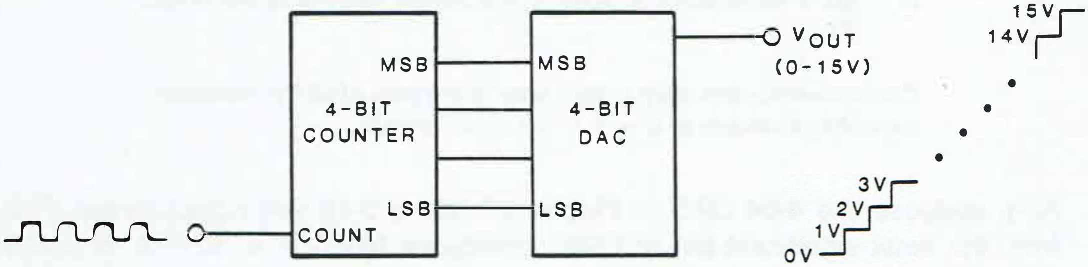


Figure 6-2
A 4-bit counter connected to the digital input lines of a 4-bit DAC will provide a staircase output from the DAC.

Let's analyze the DAC output further. Notice that each bit in the digital input word contributes an output voltage level that is proportional to its bit **position**. For example, the bit 0 position, which is the least significant bit position, or LSB, contributes an output voltage level of 1 volt. The next bit position, bit 1, contributes an output voltage level of 2 volts. Bit 2 contributes an output voltage level of 4 volts and finally bit 3, the MSB, contributes an output voltage level of 8 volts. Thus, you could conclude that the voltage contribution of a given bit position is twice that of the previous bit position. In other words, the final output voltage level is weighted according to powers of 2.

Example 6-1

What would be the output voltage of the DAC in Figure 6-2 for a digital input value of 1010?

This problem could be analyzed as follows:

a. Bit 0 contributes nothing to the output since it is cleared (logic 0).

b. Bit 1 contributes 2 volts to the output since it set (logic 1).

c. Bit 2 contributes nothing to the output since it is cleared (logic 0).

d. Bit 3 contributes 8 volts to the output since it is set (logic 1).

Consequently, the ideal output level is the sum of all the contributing voltage values, or 0 + 2 + 0 + 8 = 10 volts.

Now, suppose the 4-bit DAC in Figure 6-2 has a 0-10 volt output range. This time, the least significant bit, or LSB, contributes $10V/(2^4) = 10V/16$, or 0.625 volts. The next bit, bit 1, contributes 2×0.625 V or 1.25 volts. Bit 2 contributes 2×1.25 V, or 2.50 volts and finally the MSB, bit 3, contributes 2×2.50V, or 5.00 volts. Thus, a digital input value of 1111 would be converted to an output voltage level of $(.625) + (1.25) + (2.5) + (5.0) = 9.375$ volts. How can this be? You would think that the output level should be 10 volts for an input value of 1111. However, this is not the case since each lesser significant bit position contributes one-half the output of the previous bit position. It's like taking steps towards a wall, where each step is one-half the distance of the previous step. Theoretically, you would never get to the wall.

Now, suppose you had an 8-bit DAC with a 0-10 volt output range as shown in Figure 6-3. If you drove the DAC with an 8-bit, free-running up-counter, you would obtain a staircase output as shown in the figure. Notice that the least significant bit, or LSB, contributes $10V/(2^8) = 10V/256$, or 0.039 volts to the output. The next bit position contributes $2 \times .039$ V, or 0.078 volts, and so on until the most significant bit, or MSB, contributes 5 volts. Consequently, a digital input value of 1111 1111 is converted to an analog output value of 9.961 volts by adding all the bit-position weights. Notice from Figure 6-3 that there are 2^8, or 256 unique output voltage levels, **including zero**. Consequently, there are $2^8 - 1$, or 255, steps in the staircase output.

Of particular interest in any DAC is the contribution of the LSB. This is called the DAC **step size**, and is always calculated as follows:

$$\text{step size} = \text{rated maximum output level}/(2^n)$$

where: n is the number of input bits to the DAC.

The term step size is used because of the staircase output shown in Figure 6-3. Observe that the actual maximum output level obtained from the DAC will always differ from the rated maximum output level by the step size. In addition, there are always 2^n unique output voltage levels and $2^n - 1$ output voltage steps in a staircase output for any n-bit DAC.

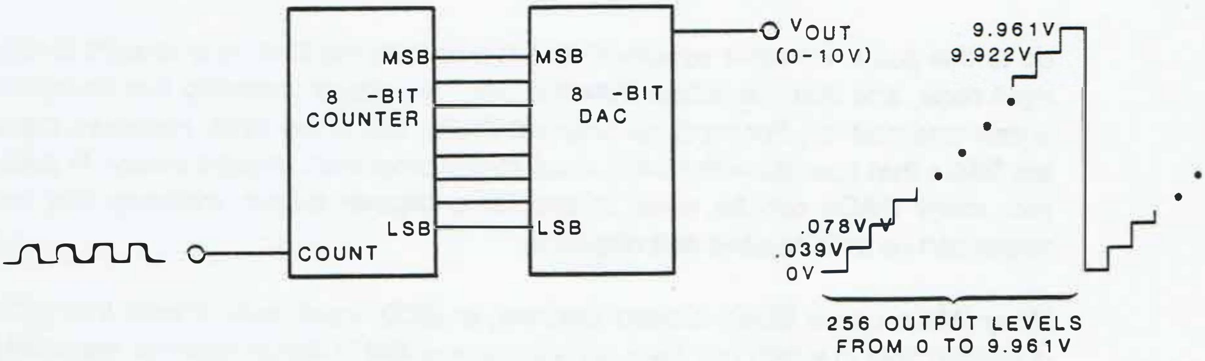


Figure 6-3

An 8-bit free running up-counter driving an 8-bit DAC.

Example 6-2

Suppose a 6-bit DAC produces an output voltage of 0.078 volts for a digital input value of 00 0001.

- a. *What is the output voltage value for a digital input value of 11 1111?*
- b. *What is the maximum rated output level for the DAC?*
- c. *How many unique output levels are there for the DAC?*

a. Obviously, 0.078 volts is the step size for this DAC. Thus, for a digital input value of 11 1111, the value of the output voltage must be 2.5 V + 1.25 V + .625 V + .3125 V + .1563 V + .078 V = 4.922 volts.

b. The step size is 0.078 volts and the actual maximum output level is 4.922 volts. Therefore, the maximum rated output level for this DAC is 4.922 V + 0.078 V = 5 V.

c. Since this is a 6-bit DAC, there are $2^6 = 64$ unique output levels, including zero. However, there are only $2^6 - 1 = 63$ steps in a staircase output for this DAC.

ADDITIONAL DAC INPUT CODES

Up to this point, we have assumed that the input to the DAC is a straight binary input code, and that the output from the DAC is unipolar, meaning that its output is only one polarity. For most commercial DACs, this is the case. However, there are DACs that operate with binary input codes other than straight binary. In addition, many DACs can be wired to provide a **bipolar** output, meaning that the output can be both positive and negative.

Many DACs use a Binary Coded Decimal, or BCD, input code. Recall that BCD is another way in which numbers can be represented in digital systems, especially in digital instrumentation. The BCD code uses 4-bit groups to represent a decimal digit value. For example, the BCD value 1001 0101 is equivalent to the decimal value 95, and the decimal value 50 would be written in BCD as 0101 0000.

Each 4-bit BCD group can range from 0000 to 1001, or 0 to 9 in decimal. Thus, an 8-bit BCD value can be used to represent decimal values between 00 and 99. The conversion table in Figure 6-4 shows the 8-bit binary and BCD equivalents for the decimal numbers 0 through 25. Notice that the binary and BCD equivalents are the same for the decimal numbers 0 through 9. However, they differ above the value 9.

Decimal	Binary	BCD
0	0000 0000	0000 0000
1	0000 0001	0000 0001
2	0000 0010	0000 0010
3	0000 0011	0000 0011
4	0000 0100	0000 0100
5	0000 0101	0000 0101
6	0000 0110	0000 0110
7	0000 0111	0000 0111
8	0000 1000	0000 1000
9	0000 1001	0000 1001
10	0000 1010	0001 0000
11	0000 1011	0001 0001
12	0000 1100	0001 0010
13	0000 1101	0001 0011
14	0000 1110	0001 0100
15	0000 1111	0001 0101
16	0001 0000	0001 0110
17	0001 0001	0001 0111
18	0001 0010	0001 1000
19	0001 0011	0001 1001
20	0001 0100	0010 0000
21	0001 0101	0010 0001
22	0001 0110	0010 0010
23	0001 0111	0010 0011
24	0001 1000	0010 0100
25	0001 1001	0010 0101

Figure 6-4

Binary and BCD equivalents for the Decimal numbers 0 through 25.

Now, suppose an 8-bit DAC with BCD inputs has a 0-10 volt output range as shown in Figure 6-5. If you connect a free-running BCD counter to the DAC input lines, you would obtain an output staircase as shown in the figure. Observe that there are 100 output values which range from 0 to 9.9 volts, with a step size of 0.1 volts. Also, the step size of the 8-bit BCD input DAC is larger than that of the 8-bit straight binary input DAC discussed previously. Thus, you might conclude correctly that a BCD input DAC cannot resolve as well as a straight binary input DAC.

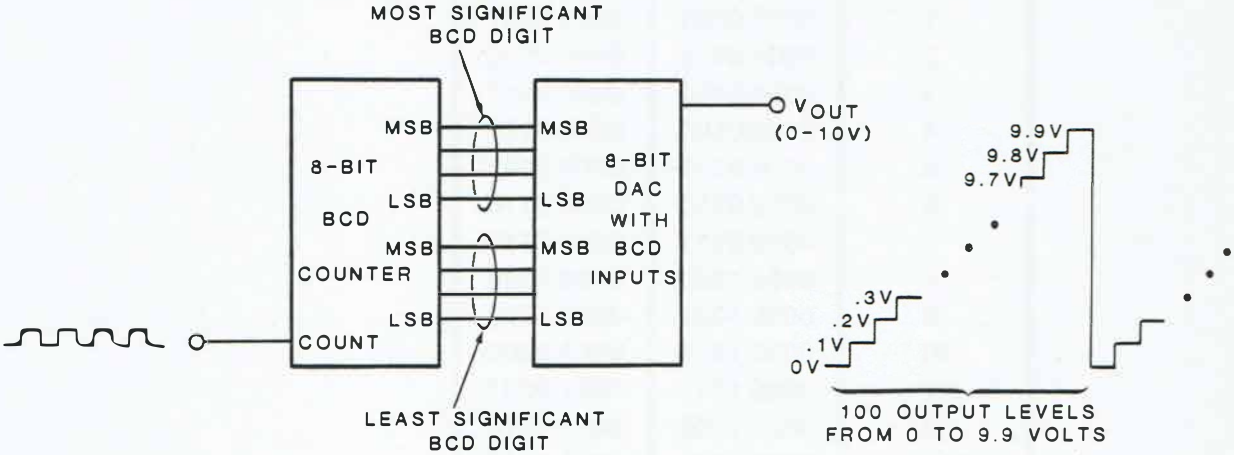

MOST SIGNIFICANT BCD DIGIT
8-BIT BCD COUNTER
MSB
LSB
MSB
LSB
COUNT
MSB
LSB
MSB
LSB
8-BIT DAC WITH BCD INPUTS
VOUT (0-10V)
9.9V
9.8V
9.7V
.3V
.2V
.1V
0V
LEAST SIGNIFICANT BCD DIGIT
100 OUTPUT LEVELS FROM 0 TO 9.9 VOLTS

Figure 6-5

An 8-bit DAC with BCD input lines will generate 100 unique output levels which are proportional to the BCD input value.

Other binary input codes are used with **bipolar** DACs to provide both positive and negative output levels which correspond to positive and negative, or signed, binary input values. Three binary codes are common — **sign magnitude binary, two's complement binary,** and **offset binary**. A comparison table for a ±5-volt, 4-bit DAC with these different input codes is shown in Figure 6-6. Let's briefly examine each input code as it relates to the given output voltage value.

Sign magnitude is probably the simplest to understand. The sign, + or −, is determined by the most significant bit, or MSB. A zero indicates a positive number and a one indicates a negative number. The magnitude of the number is then interpreted by the straight binary weight of the remaining bits.

Two's complement binary is the most popular signed code since most microprocessors use this code to perform arithmetic operations. Recall from past experience, or from reading Appendix A, that the two's complement of a binary value is found by taking its one's complement and adding 1. Thus, the 4-bit two's complement of $+7_{10}$, or 0111_2, is 1000 + 1 = 1001.

Decimal Equivalent	DAC Input Code			Bipolar DAC Output
	Sign and Magnitude	Two's Complement	Offset Binary	V_{OUT}
+7	0111	0111	1111	+4.375 V
+6	0110	0110	1110	+3.75 V
+5	0101	0101	1101	+3.125 V
+4	0100	0100	1100	+2.50 V
+3	0011	0011	1011	+1.875 V
+2	0010	0010	1010	+1.25 V
+1	0001	0001	1001	+ .625 V
+0	0000	0000	1000	0 V
−0	1000	0000	1000	0 V
−1	1001	1111	0111	− .625 V
−2	1010	1110	0110	−1.25 V
−3	1011	1101	0101	−1.875 V
−4	1100	1100	0100	−2.50 V
−5	1101	1011	0011	−3.125 V
−6	1110	1010	0010	−3.75 V
−7	1111	1001	0001	−4.375 V
−8	XXXX	1000	0000	−5.0 V

Figure 6-6
Signed input codes used with bipolar DACs.
Here a + 5 volt 4-bit bipolar DAC.

In two's complement binary code, the negative binary representation of a value is found by taking its two's complement. Consequently, $0111_2 = +7_{10}$ and its two's complement, $1001_2 = -7_{10}$. Again, the MSB is used to denote the sign of the value. A zero denotes a positive number and a one denotes a negative number.

Offset binary is not as frequently used as two's complement binary; however, it is very similar. Notice from Figure 6-6, that offset binary is identical to two's complement binary, except that the sign bit, or MSB, is inverted.

Most of the time the DAC will use a straight binary input, and provides a unipolar output. However, many commercial DACs can be wired to provide a bipolar output. There will be times when the application dictates bipolar operation. When this is the case, the two's complement input code is most often used by the DAC.

DAC SPECIFICATIONS

There are two major specifications associated with any DAC that will be important to you. They are **resolution** and **accuracy**.

The most important specification is resolution.

> **resolution:** The smallest analog output change that can occur as a result of a change in the digital input.

Manufacturers state resolution in several different ways. The simplest is by the number of DAC input bits. For example, an 8-bit DAC has an 8-bit resolution, a 10-bit DAC has a 10-bit resolution, and so on. Another way to state resolution is to say that an 8-bit DAC has a resolution of 1-part-in-256, since $2^8 = 256$. In the same way, a 10-bit DAC has a resolution of 1-part-in-1024, since $2^{10} = 1024$. However, a more meaningful way to express resolution is as a percentage of the maximum output of the DAC.

Recall that resolution indicates the smallest output change that can occur as a result of an input change. The smallest input change that can occur is a change in the least significant input bit, or LSB. The LSB has a weight of $1/2^n$, where n is the number of digital input bits to the DAC. Thus, you can calculate the % resolution of any DAC as follows:

$$\% \text{ resolution} = 1/2^n \times 100\%$$

Example 6-3

Calculate the % resolution of the following DACs:

a. 8-bit DAC.
b. 10-bit DAC.

a. For an 8-bit DAC:

$$\begin{aligned}\%\text{ resolution} &= 1/2^8 \times 100\% \\ &= 1/256 \times 100\% \\ &= 0.39\%\end{aligned}$$

b. For a 10-bit DAC:

$$\begin{aligned}\%\text{ resolution} &= 1/2^{10} \times 100\% \\ &= 1/1024 \times 100\% \\ &= 0.0976\%\end{aligned}$$

A resolution table for different n-bit DACs is shown in Figure 6-7. In addition to showing the % resolution, Figure 6-7 shows the resolution, in volts, for the given DACs with a 0-10 V output range. Since the resolution of a DAC is directly related to the LSB weight, a DAC with a given output range will always have a resolution equal to its step size. As you can readily see, the more input bits a DAC has, the better its resolution.

Here's why resolution is so important when designing control circuits. Many control applications require the measurement and display of revolutions per minute, or rpm. Suppose a digital circuit is being used to measure the rpm of a shaft and provide the operator with an analog indication of the rpm via a tachometer. Suppose also that the maximum speed of the shaft is 1000 rpm and a DAC is used to drive the tachometer movement.

DAC	% resolution	Resolution with a 0-10 V Output Range
4-bit	6.25	625 millivolts
6-bit	1.56	156 millivolts
8-bit	.39	39 millivolts
10-bit	.0976	9.76 millivolts
12-bit	.0244	2.44 millivolts
14-bit	.0061	.61 millivolts

Figure 6-7
A resolution comparison table for different DACs.

If a 4-bit DAC is used to drive the tachometer, the rpm could only be resolved to $1/2^4 \times 1000$, or 62.5 rpm as shown in Figure 6-8a. This is definitely not precise enough for most control applications. The operator would have to guess the rpm between increments of 62.5. However, if an 8-bit DAC were used, the rpm could be resolved to $1/2^8 \times 1000$, or 3.9 rpm as shown in Figure 6-8b. This still might not be precise enough for the control application. However, if a 10-bit DAC were used as shown in Figure 6-8c, the rpm could be resolved to $1/2^{10} \times 1000$, or 0.976 rpm which is probably good enough for many control applications.

The point is this: a DAC with more input bits provides better resolution; however, the **application will always dictate the design**. You would not use a 10-bit DAC in the tachometer example if the application dictates that you only need to resolve to the nearest 10 rpm. An 8-bit DAC would be more than enough to do the job.

The second important DAC specification is accuracy.

> **accuracy**: The actual analog output obtained from a DAC compared to the expected output.

Accuracy is sometimes called **linearity**. Like resolution, accuracy can be expressed as a percentage of the maximum output of the DAC. For example, if a 0-10 volt DAC has an accuracy of ±0.19%, the actual output will be within $\pm(0.0019 \times 10\text{ V})$, or ±19 millivolts of the expected value.

Another way that manufacturers specify accuracy is related to the LSB weight of the DAC. For instance an 8-bit DAC specification might indicate that the DAC accuracy is ±1/2 LSB. The LSB has a weight of $1/2^n$. Thus, the accuracy for this DAC must be $\pm 1/2\ (1/2^8) \times 100\%$, or ±0.19%. With a 0 – 10-volt output range, the accuracy, in volts, of this DAC would be $\pm(0.0019 \times 10\text{ V})$, or ±19 millivolts. You might note that a DAC with an accuracy of ±1/2 LSB has an accuracy which is equal to 50% of its resolution, or step size. Most commercial DACs have an accuracy of at least ±1/2 LSB.

Another less important specification for a DAC is settling time.

> **settling time:** The time it takes for a changing DAC output to settle to 99.95% of its new value.

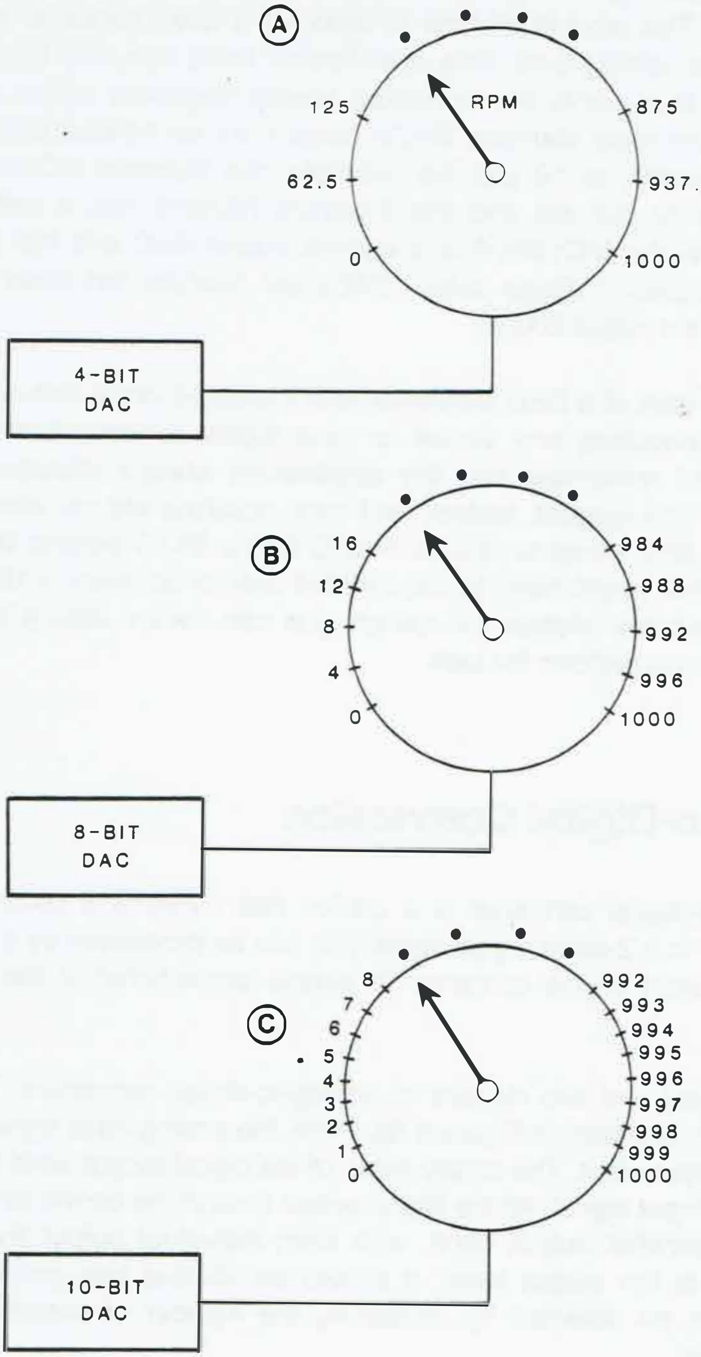


Figure 6-8

A 0-1000 RPM tachometer resolution with various DACs.

Each digital-to-analog conversion takes time since no process is immediate or instantaneous. The amount of time it takes for a DAC output to stabilize to a new value is its settling time. This specification limits how fast conversions can be made and, thus, limits the maximum analog frequency output of the DAC. Settling times for most standard DACs range from 50 nanoseconds, or 50 nS, to 10 microseconds, or 10 µS. For example, the Motorola MC1408 DAC has a settling time of 300 nS and the Signetics NE5018 has a settling time of 2 µS. However, the MC1408-8 is a current output DAC and the NE5018 is a voltage output DAC. Voltage output DACs are typically ten times slower than equivalent current output DACs.

Of course, the cost of a DAC increases with increased resolution, accuracy, and speed. When choosing any device for your digital control system, such as a DAC, you must remember that the **application always dictates** the device specifications. The biggest, fastest, and most accurate are not always the best. It would make little sense to require a DAC with a 50 nS settling time to control a thermostat that might need to be updated only once every 5-10 minutes. By letting the application dictate the design, you can always assure the most cost effective system to perform the task.

Analog-To-Digital Conversion

The analog-to-digital converter is a device that converts a continuous analog voltage signal to a 2-state digital signal that can be processed by a digital circuit. The digital output of the converter is always proportional to the analog input value.

In general, there are two classes of analog-to-digital converters. The first, and most common, is shown in Figure 6-9a. Here, the analog input signal is converted to a digital output word. The binary value of the digital output word is proportional to the analog input signal. All the digital output lines of the converter are combined to form the **parallel** output word, with each individual output line contributing one data bit to the output word. It should be obvious that greater conversion resolution can be obtained by increasing the number of output lines, or bits, of the converter.

Another class of analog-to-digital converter is shown in Figure 6-9b. This device converts an analog input signal into a train of serial digital output pulses. The rate at which the digital output pulses are generated is proportional to the analog input voltage level. Consequently, the digital output frequency of this converter is proportional to its analog input voltage level. This type of converter is commonly referred to as a **voltage-to-frequency**, or **V/F, converter**.

Some clarification is needed at this point. It is common practice and accepted throughout the industry to refer to the serial output device shown in Figure 6-9b as a V/F converter and not an ADC, even though it is an analog-to-digital converter. The designation "ADC" is generally associated with the parallel output device shown in Figure 6-9a. So, from now on, we will use this accepted terminology.

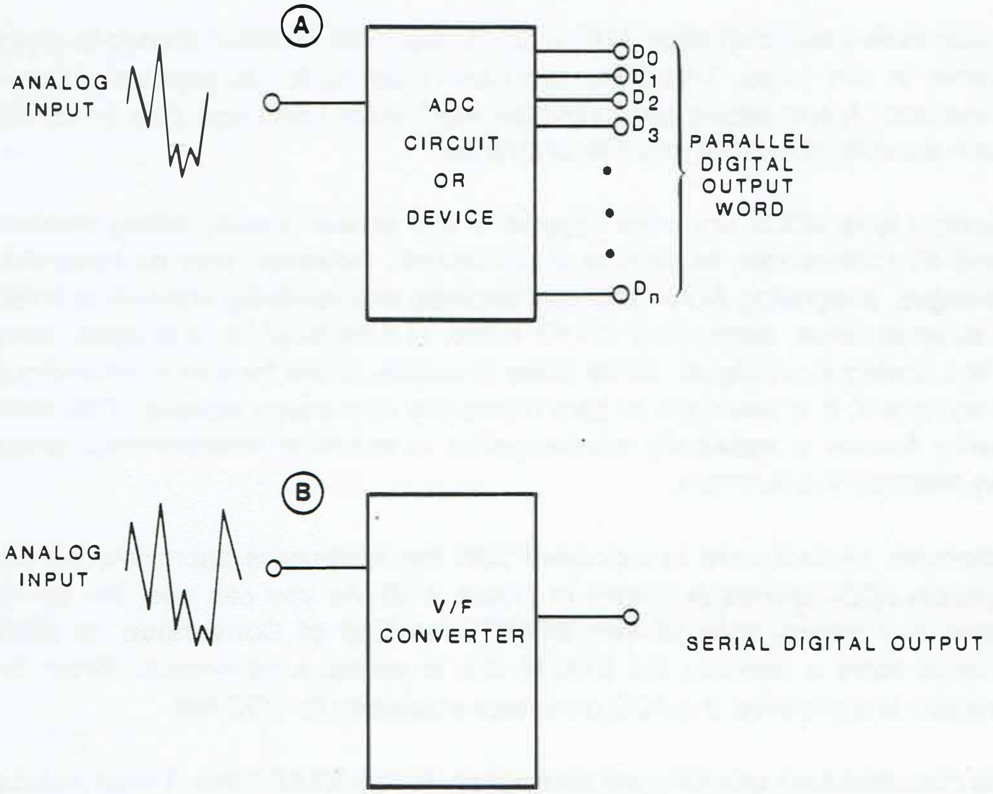


Figure 6-9

Two general types of analog-to-digital converters: a parallel output ADC (a), and a serial output V/F converter (b).

ADCs

There are basically two types of parallel-output ADC devices commercially available. They are the **successive approximation** and **integration** type ADCs. Both types perform the same analog-to-digital conversion function, but use different principles of operation. A detailed study of how they operate is beyond the scope of this course. However, you should know the general characteristics of these devices so that you can pick a given device for a given application.

The successive approximation ADC is by far the most common analog-to-digital converter in use today. There are two basic reasons for its popularity: speed and low cost. A successive approximation ADC which costs less than $4.00 can perform a conversion in the microsecond range.

Integration type ADCs are more expensive and slower, usually taking between 10 and 50 milliseconds to perform a conversion. However, they do have their advantages. Integrating ADCs are very accurate and relatively immune to noise. Any external noise, particularly 60 Hz noise, is integrated, or averaged, along with the analog input signal. Since noise is usually in the form of a symmetrical sine waveform, it is averaged to zero during the conversion process. This noise immunity feature is especially advantageous in industrial environments, where heavy machinery is common.

A schematic symbol used to represent both the successive approximation and integration ADC devices is shown in Figure 6-10. As you can see, the device contains two control lines labeled **START** and **End of Conversion**, or **EOC**. You must apply a pulse to the START line to initiate a conversion. When the conversion is completed, the ADC generates a pulse on its EOC line.

Some manufacturers use different terminology for the START line. These include: Initiate Conversion (IC), BEGIN, RUN, CONVERT, etc. However the idea is the same: when your digital circuit activates this line, the ADC begins to convert the unknown analog input signal. In this way, your digital circuit can control the conversion process.

The End of Conversion, or EOC, control line is sometimes called Data Valid (DV), DONE, STATUS, or BUSY. In all cases, this line is an output control line that is activated by the ADC to signal when the conversion process is complete. In other words, the ADC generates an active logic transition on this line each time it completes a conversion.

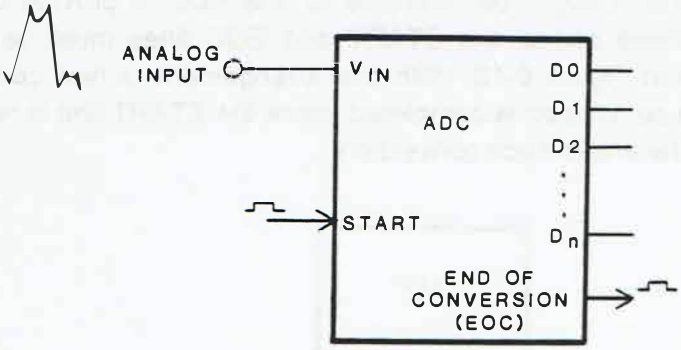


Figure 6-10
A schematic symbol for a successive approximation or integration type ADC.

The converted digital output value appears on the ADC output lines when the EOC line makes an active transition. Most ADCs have internal output latches. These output latches hold the converted value until another conversion is made. However, if the ADC output is not latched, the EOC line must be used to activate the LOAD line on a buffer register, as shown in Figure 6-11, to latch the output.

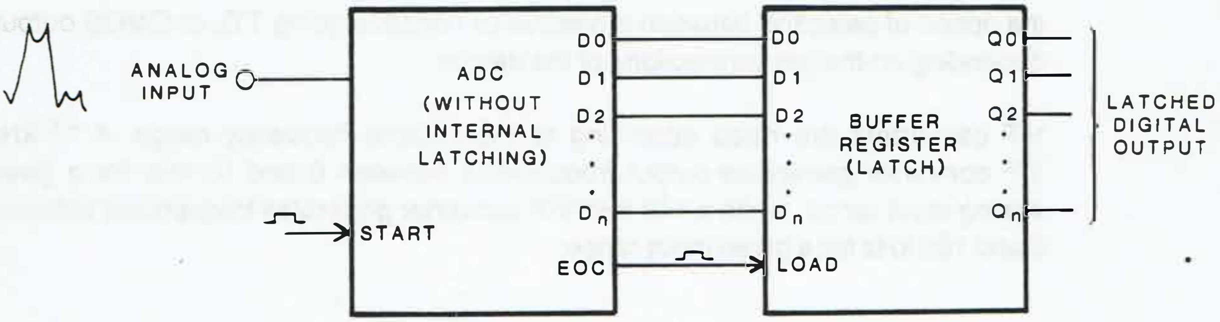


Figure 6-11
An external latch must be used if the ADC outputs are not internally latched.

In some applications, it might be desirable for the ADC to provide continuous conversions. For these cases, the START and EOC lines must be jumpered together as shown in Figure 6-12. With this arrangement, a new conversion is started each time a conversion is completed, since the START line is re-activated by the EOC line at the end of each conversion.

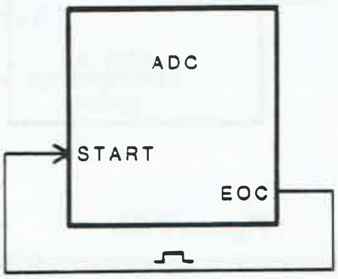


Figure 6-12
Continuous conversions are possible by jumpering the START and EOC control lines.

V/F CONVERTERS

The output of a V/F converter is a series of square-wave pulses that are compatible with either TTL or CMOS logic as shown in Figure 6-13. You usually have the option of selecting between a positive or negative going TTL or CMOS output, depending on the pin connections of the device.

V/F converters are rated according to their output frequency range. A 10 kHz V/F converter generates output frequencies between 0 and 10 kHz for a given analog input range, while a 100 kHz V/F converter generates frequencies between 0 and 100 kHz for a given input range.

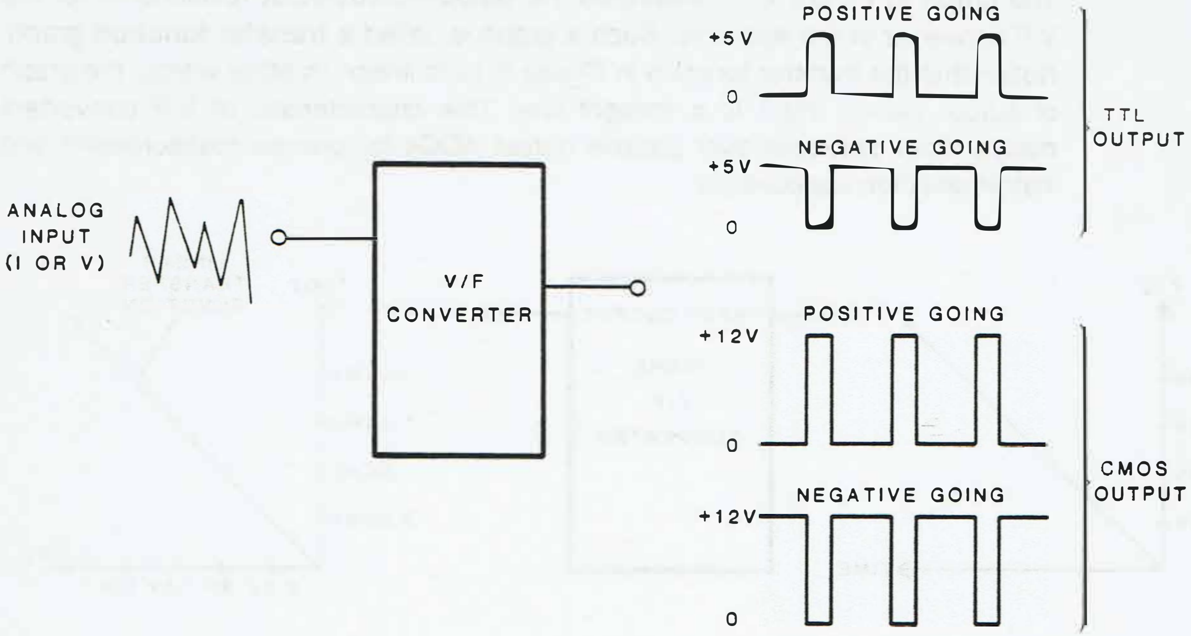


Figure 6-13

V/F converter outputs are usually compatible with both TTL and CMOS logic.

Example 6-4

Suppose that the 10 kHz V/F converter shown in Figure 6-14 has an input voltage range of 0 to +10 volts. Determine the output frequency for input values of 10 volts, 7.5 volts, and 5 volts.

From the associated graph you can see that a 10-volt input will generate an output frequency of 10 kHz. Likewise, a 7.5-volt input level is converted to a 7.5 kHz frequency, and a 5-volt input level is converted to a 5 kHz frequency.

The graph in Figure 6-14 illustrates the output-versus-input relationship for the V/F converter in the example. Such a graph is called a **transfer function** graph. Notice that the transfer function in Figure 6-14 is linear. In other words, the graph of output versus input is a straight line. This characteristic of V/F converters makes them desirable over parallel output ADCs for precise measurement and instrumentation applications.

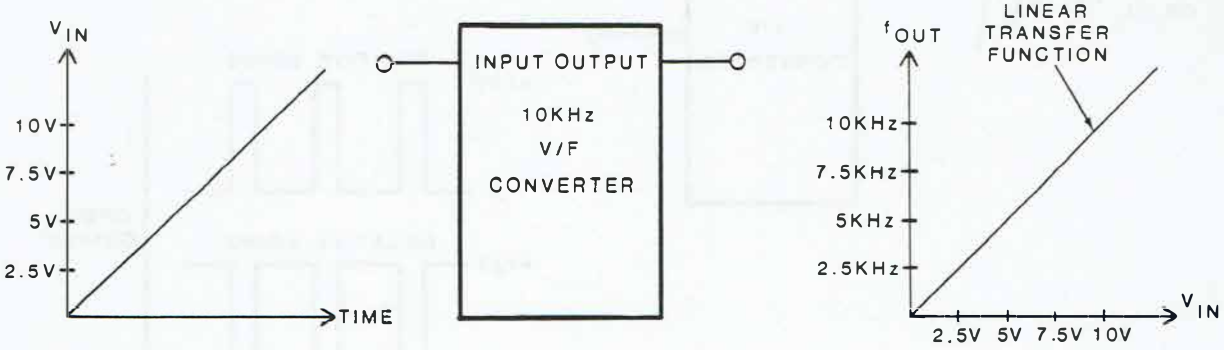


Figure 6-14

The output of a V/F converter is very linear as shown by its transfer function.

To obtain the digital equivalent of an analog input signal, you must count the number of pulses generated by the V/F converter in a given time period. This is shown in Figure 6-15. Here, the V/F converter output is connected to the input of a binary counter. An external signal (0.5 Hz) is used to enable the counter for 1 second. After the 1 second period, the counter contains the binary equivalent of the output frequency. Thus, the count is directly proportional to the analog signal level.

Other desirable characteristics of V/F converters for industrial use are their high temperature stability and noise rejection ability. Temperature stability for these devices ranges from 10 to 100 parts per million (ppm) per degree centigrade (°C), depending on the device. Also, since the V/F converter is essentially an integrating device it has excellent noise immunity, similar to that of integration type ADCs.

Like integrating type ADCs, the integrating process used by V/F converters makes them relatively slow compared to successive approximation ADCs. However, they are fast enough for many industrial applications such as temperature and pressure measurement.

Even though V/F converters are not as widely used as parallel output ADCs, they are becoming more popular, especially for remote sensing applications in an industrial environment. Also, many V/F converter devices can be used as frequency-to-voltage, or F/V converters. When this is the case, the device is sometimes called a **V/F/V converter**.

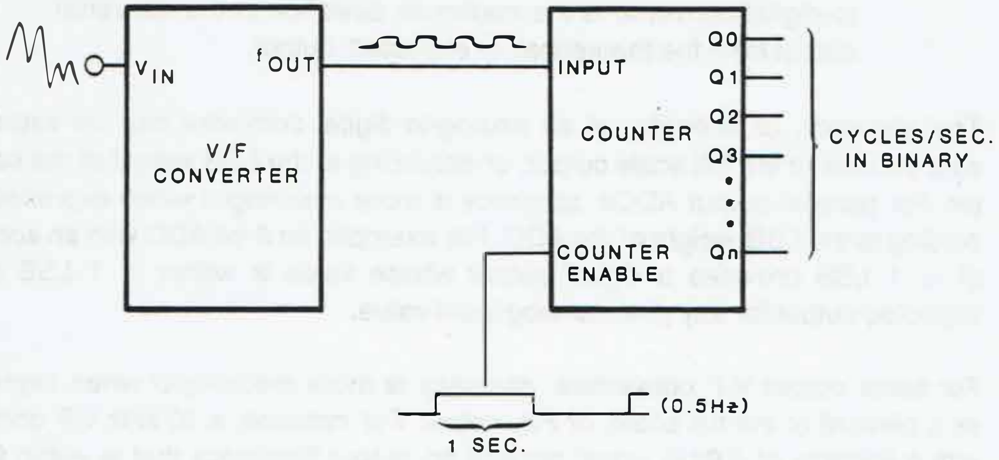


Figure 6-15

A counter must be used to count the output pulses of a V/F converter per unit time.

SPECIFICATIONS AND COMPARISONS

The three most descriptive specifications of an analog-to-digital converter are **resolution, accuracy (linearity)**, and **conversion time**.

> **resolution**: The smallest analog input change that can be distinguished by the analog-to-digital converter.

Even though the definition for analog-to-digital converter resolution relates to input changes, it is usually specified according to the converter output. For parallel output ADCs, resolution is expressed as the number of output bits. Thus, you say that an 8-bit ADC has an 8-bit resolution.

For V/F converters, resolution is expressed by the maximum output frequency and measurement period. If the measurement period of a 10 kHz V/F converter is one second, its resolution would be one part in 10,000. On the other hand, if the measurement period is one tenth of one second, or 0.1 seconds, its resolution would be 1 part in 1000.

> **accuracy (linearity)**: The accuracy, or linearity, of an analog-to-digital converter is the maximum deviation of the converter output from the theoretical, or expected, output.

The accuracy, or linearity, of an analog-to-digital converter can be expressed as a percent of the full scale output, or according to the LSB weight of the converter. For parallel output ADCs, accuracy is more meaningful when expressed according to the LSB weight of the ADC. For example, an 8-bit ADC with an accuracy of ± 1 LSB provides a digital output whose value is within ± 1 LSB of the expected output for any given analog input value.

For serial output V/F converters, accuracy is more meaningful when expressed as a percent of the full scale, or FS, output. For instance, a 10 kHz V/F converter with a linearity of 0.01% would provide an output frequency that is within 0.01% of 10 kHz. This equates to an actual output that is within 1 Hz of the expected output frequency for any given analog input value.

> **conversion time**: The time it takes for an analog-to-digital converter to perform a single conversion to the specified resolution and linearity for a full scale analog input change.

The meaning of this specification is fairly obvious. However, note that it is the conversion time required for a **full-scale** analog input change. Consequently, the specified conversion time is the worst case, or maximum, amount of time required for any conversion performed by the ADC. Furthermore, the conversion time for a V/F converter is equal to the measurement period. As you are aware, the measurement period of a V/F converter is controlled by the user.

Finally, the table in Figure 6-16 provides relative comparisons of the analog-to-digital converters discussed in this section. You will not usually find it hard to decide what type of converter is best for your application, since the application will always dictate the best choice.

CHARACTERISTIC	SUCCESSIVE APPROXIMATION	INTEGRATION	V/F CONV.
ACCURACY (LINEARITY)	GOOD	VERY GOOD	EXCELLENT
CONVERSION TIME	VERY FAST	SLOW	VERY SLOW
NOISE REJECTION	POOR	EXCELLENT	EXCELLENT
TEMP. STABILITY	GOOD	VERY GOOD	EXCELLENT

Figure 6-16
Analog-to-digital converter comparisons.

Self-Test Review

1. Calculate the following for a 6-bit DAC with a 0 to 10 volt output range.

   A. Step size = ______________
   B. % resolution = ______________

2. Suppose the DAC in question 1 has an accuracy of ± 1/2 LSB. How close will the actual DAC output be to the expected output value?

   ____
   ____

3. Describe the difference between a unipolar and bipolar DAC output.

   ____
   ____

4. The offset binary value of 0000 0000 is equivalent to a two's complement binary value of ________________.

5. How many output steps would there be for a 10-bit DAC? ______________

6. Name the two general classes of analog-to-digital converters.

   A. ____

   B. ____

7. Another name for the accuracy of an ADC is ______________.

8. Describe the operation of a parallel output ADC. ____________________

____

____

____

9. What is the accuracy of a 100 kHz V/F converter if its output is within 10 Hz of its expected output? ____________________

10. Explain how to convert the serial output pulses of a V/F converter into a binary value that is proportional to the unknown analog input.

____

____

____

Answers

1A. 0.156 volts

1B. 1.56%

2. ±0.078 volts

3. A unipolar DAC output is only one polarity, whereas a bipolar DAC output can be positive or negative, depending on the DAC input value.

4. 1000 0000

5. $2^{10} - 1$, or 1023

6A. Parallel output ADC

6B. Serial output V/F converter

7. linearity

8. A parallel output ADC starts the conversion process when its START line is activated. When the conversion is complete, it generates an active transition on its EOC line. At this time, the converted digital value will appear on the ADC output lines.

9. 0.01%

10. The serial output pulses of a V/F converter must be counted. The accumulated count in a given time period is proportional to the unknown analog input value (See Figure 6-15).

Following is the solution to Questions 1, 2, and 9.

1A. Step size $= 10\text{V}/(2^6) = 10\text{V}/64 = 0.156$ V

1B. % resolution $= 1/(2^6) \times 100\% = 1.56\%$

2. $\pm 1/2$ LSB accuracy $= \pm 1/2 \times 1.56\% = \pm 0.78\%$
Thus, the DAC in question 1 would have an actual output value within ±(0.0078 times 10 V), or ±0.078 volts of the expected output value.

9. (10 Hz/100 kHz) x 100% = 0.01%

Before you read the next section, go to Unit 8 and perform Experiments 15 and 16.

THE POWER SUPPLY

One of the primary support circuits in any digital design is the power supply. Clearly, nothing will work without one, but less obvious is the fact that many digital circuits are designed and marketed with an inadequate power supply. At a minimum, the power supply must do exactly what it says: supply adequate power to operate the circuit. To do this, you, the circuit designer, must determine how much power is required by the circuit in order to specify the appropriate supply.

I do not intend to teach you how to design power supplies in this course. Such a topic is better suited to a course in analog circuits. Furthermore, it makes no sense to design your own supply when there are a multitude of inexpensive power supply modules commercially available to satisfy just about any power requirement. The intent of the following discussion is to aquaint you with the features of a power supply and show you how to determine the power requirements of your design. You will then use this information to specify an adequate commercial power supply module.

Power Supply Basics

All digital systems require a source of DC power. In portable CMOS circuits, the power supply may simply consist of one or more batteries. Batteries, however, do not supply enough power to operate TTL circuits for long periods of time, let alone, continuously. Even with CMOS circuits, batteries must be replaced periodically. Consequently, when a source of AC power is available, it is usually more economical to convert the AC power to DC power for circuit operation. This is the job of the power supply module. Aside from batteries, there are basically two types of power supplies found in digital circuits: the **linear supply** and the **switching supply**.

LINEAR SUPPLIES

A block diagram of a typical linear power supply circuit is shown in Figure 6-17. The circuit input is usually standard household power: 110 V, 60 Hz AC power. The function of the power supply is to convert this AC power to the DC levels required by the digital circuit. On the input side of the supply, a **power transformer** steps-down the 110-volt line voltage to a voltage level nearer to that required by the digital circuit. A **diode rectifier** circuit then converts the bipolar AC voltage to a unipolar pulsating DC voltage as shown. The pulsating DC voltage is smoothed-out by a **smoothing filter** network, and passed-on to a **regulator circuit**.

As far as the digital circuit, or load, is concerned, the regulator is one of the most important parts of the supply. The job of the regulator is to regulate, or control, the output voltage at a relatively constant level, regardless of the changing load conditions. In other words, the regulator must supply a constant voltage level to the digital circuit under normal operating conditions to insure proper operation. Poor power regulation is one of the most frequent problems encountered in digital circuits.

Finally, the regulated DC voltage is sometimes filtered to eliminate any remaining AC ripple. Additional circuitry may be found in more sophisticated power supplies, but the fundamental blocks shown in Figure 6-17 are common to most supplies.

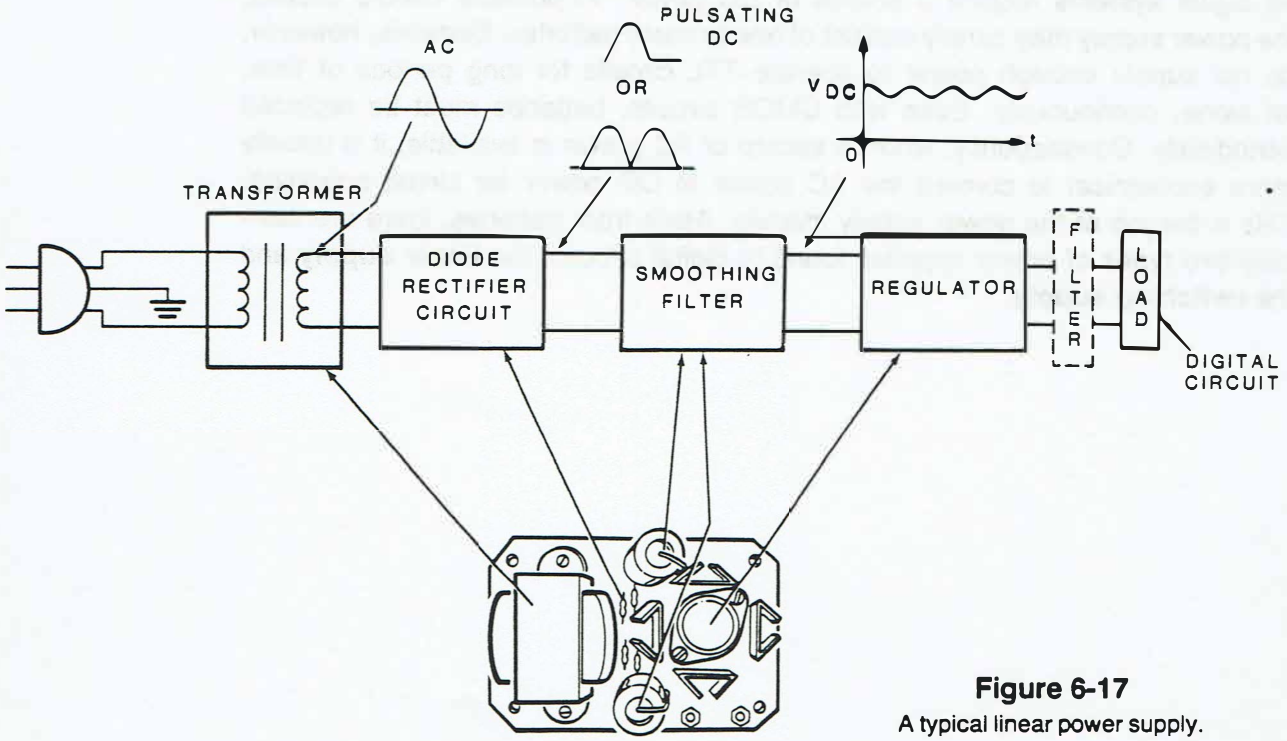


Figure 6-17

A typical linear power supply.

In a linear power supply, regulation is usually provided by placing a transistor in series or parallel with the load as shown in Figure 6-18. In either case, the transistor acts like a variable resistance. The regulator automatically compensates for any input voltage or load changes by varying its resistance to maintain a constant output level. Let's take a closer look at the operation of the series regulator in Figure 6-18a.

Suppose the power supply must generate a constant +5-volt output level, and provide a maximum output current of 1 ampere for a TTL circuit. Also, suppose that the input to the regulator in Figure 6-18a is +16 volts. Then, to supply the required +5-volt level, the regulator must **drop** 16 V − 5 V, or 11 volts. Now, if the TTL load circuit draws the full 1 ampere of current, the power dissipated by the regulator is 11 V × 1 A, or 11 watts (W). This power is dissipated in the form of heat by the regulator transistor and is essentially wasted. More or less power is dissipated, depending on the regulator input voltage and load current. This power dissipation represents a direct loss to the system, resulting in decreased efficiency. The solution to the inherent inefficiency of a linear power supply is found in a switching power supply.

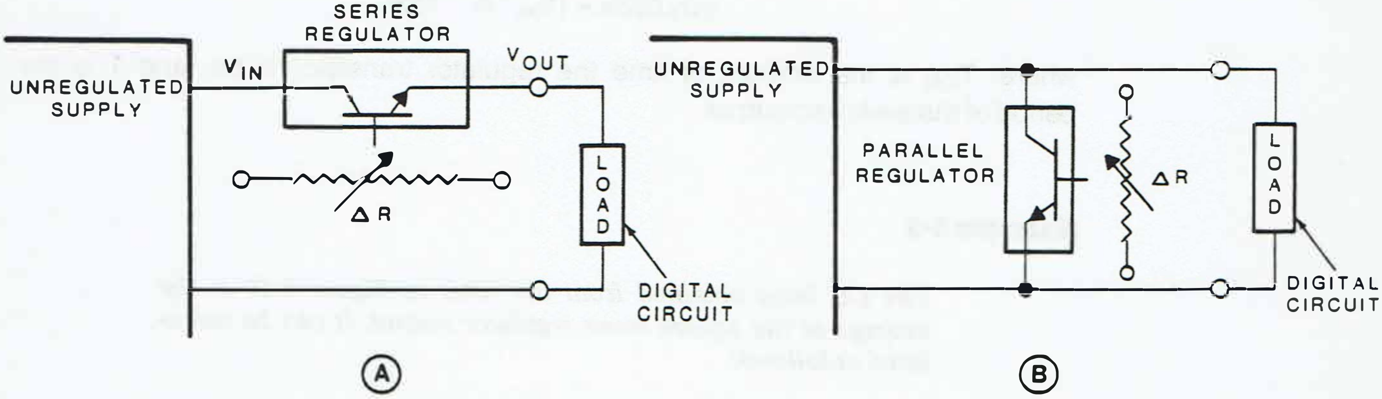


Figure 6-18
A series regulator (a) and a parallel regulator (b) for an unregulated linear power supply.

SWITCHING SUPPLIES

The major difference between a linear and switching supply is the regulator circuit. In a switching supply the regulator transistor is operated as a switch, rather than a variable resistor, as shown in Figure 6-19. Here, the transistor is turned on and off at a high rate by an associated control circuit. As a result, the transistor "chops-up" the input voltage into a series of square wave pulses as shown. These pulses are then filtered by an LC (inductance/capacitance) filter to produce a constant DC level that is proportional to the duty cycle and frequency of the square wave pulses. Since no voltage is being dropped by the switching transistor, a switching supply is very efficient, approaching 95%.

Most switching regulators operate around a constant 20 kHz switching rate. The control circuit monitors the load conditions and varies the duty cycle of the regulator output to maintain a constant output voltage level. This is done by changing the amount of time the transistor is on in any given cycle.

Recall that the duty cycle of a square waveform is the ratio of high, or on, time to the total period, or:

$$\text{Duty Cycle} = (T_{ON} / T) \times 100\%$$

where: T_{ON} is the amount of time the regulator transistor is on, and T is the period of the switched output.

Example 6-5

The DC level obtained from the filter in Figure 6-19 is the average of the square wave regulator output. It can be calculated as follows:

$$\text{DC Output} = (V_{ON}\ \text{level} \times \text{Duty Cycle})/100\%$$

Assume that the transistor is switching at a 20 kHz rate and is on for 10 μs of each cycle. Also, suppose the V_{ON} level is 25 volts.

a. *Calculate the regulator duty cycle.*
b. *Calculate the DC voltage level produced by the supply.*

a. First, since the transistor is switching at a 20 kHz rate, it is generating a square waveform with a period of:

$$\begin{aligned} \text{Period} &= 1/20\ \text{kHz} \\ &= 1/(20 \times 10^3\ \text{Hz}) \\ &= 0.05 \times 10^{-3}\ \text{S} \\ &= 50\ \mu\text{S} \end{aligned}$$

Since the transistor on-time is 10 μs, the duty cycle is:

$$\begin{aligned} \text{Duty Cycle} &= (10\ \mu\text{S}/50\ \mu\text{S}) \times 100\% \\ &= 20\% \end{aligned}$$

b. With a V_{ON} level of 25 volts, the duty cycle calculated in Part A produces a DC output level of:

$$\begin{aligned} \text{DC Output} &= (25\ \text{V} \times 20\%)/100\% \\ &= 5\ \text{V} \end{aligned}$$

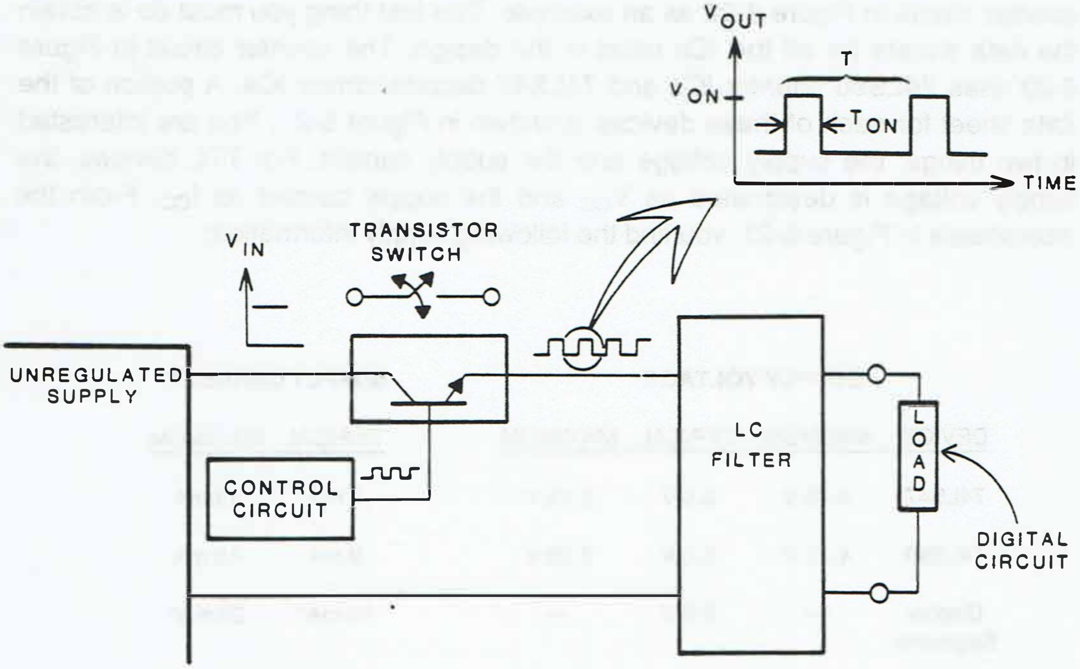


Figure 6-19
A switching regulator.

Determining the Circuit Power Requirements

Many digital circuits suffer from an inadequate power supply. This occurs not only in the cost-conscious consumer electronic field, but also in industrial, biomedical, and educational electronics. The two most common problems are inadequate current sourcing capability and inadequate voltage regulation. In many cases, the reason for these inadequacies is that the circuit power requirements are determined too early in the circuit design process, and not re-evaluated after the design has gone through several refinements and evolutionary changes. Clearly, any permanent decisions regarding the selection of a power supply must be postponed until the rest of the circuit design is finalized. This is why I have delayed a discussion of power supplies until this point in the text.

Now that you have been acquainted with power supplies, it is time to determine the power requirements of the digital circuit to be supplied. Let's use the LED counter circuit in Figure 6-20 as an example. The first thing you must do is obtain the data sheets for all the ICs used in the design. The counter circuit in Figure 6-20 uses 74LS90 counter ICs and 74LS47 decoder/driver ICs. A portion of the data sheet for each of these devices is shown in Figure 6-21. You are interested in two things: the supply voltage and the supply current. For TTL devices, the supply voltage is designated as V_{CC} and the supply current as I_{CC}. From the data sheets in Figure 6-21, you find the following supply information:

	SUPPLY VOLTAGE			SUPPLY CURRENT	
DEVICE	MINIMUM	TYPICAL	MAXIMUM	TYPICAL	MAXIMUM
74LS47	4.75 V	5.0 V	5.25 V	7 mA	13 mA
74LS90	4.75 V	5.0 V	5.25 V	9 mA	15 mA
Display Segments	—	5.0 V	—	10 mA*	20 mA*

* Requires current limiting resistors.

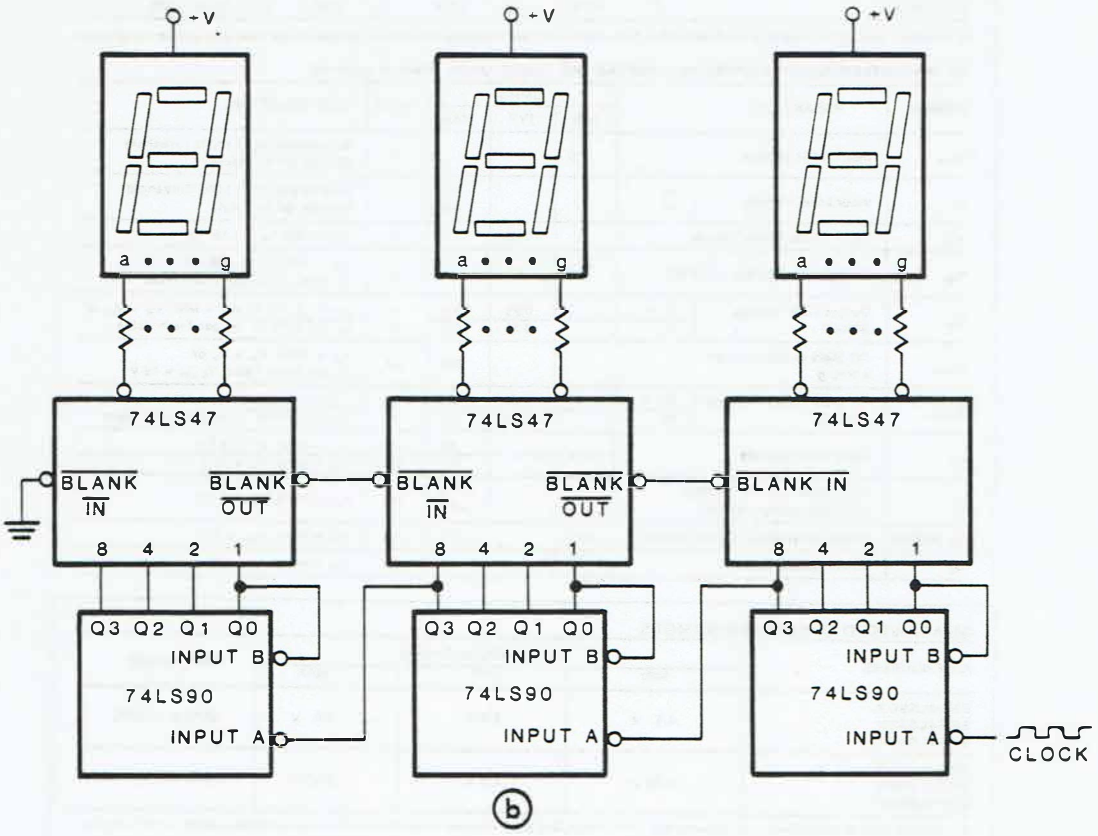
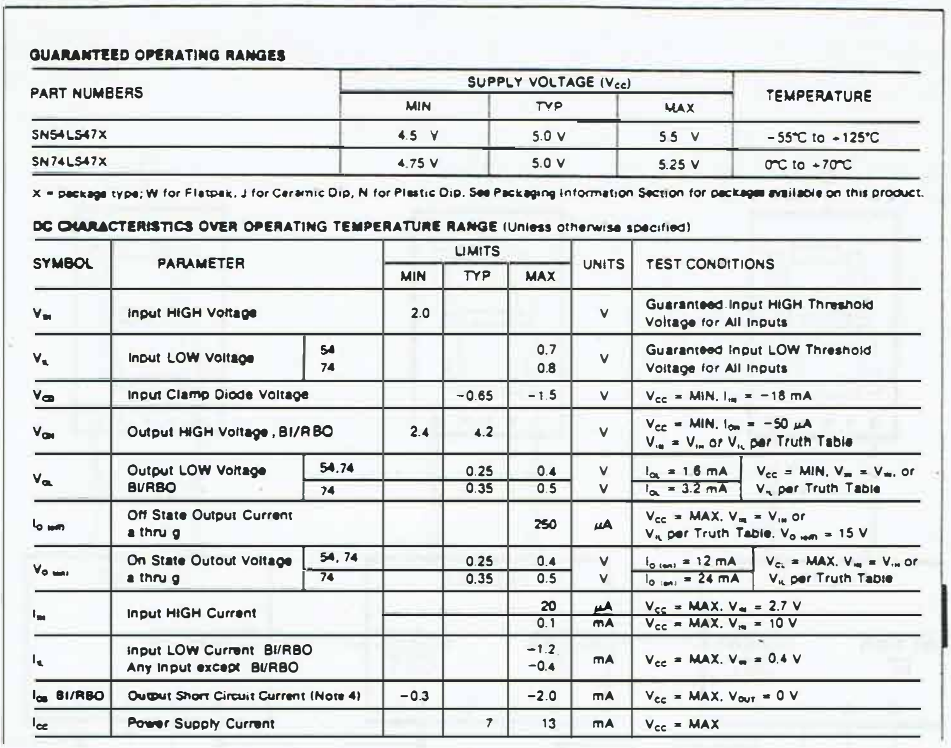


Figure 6-20

A decimal counting circuit.

(a)



GUARANTEED OPERATING RANGES

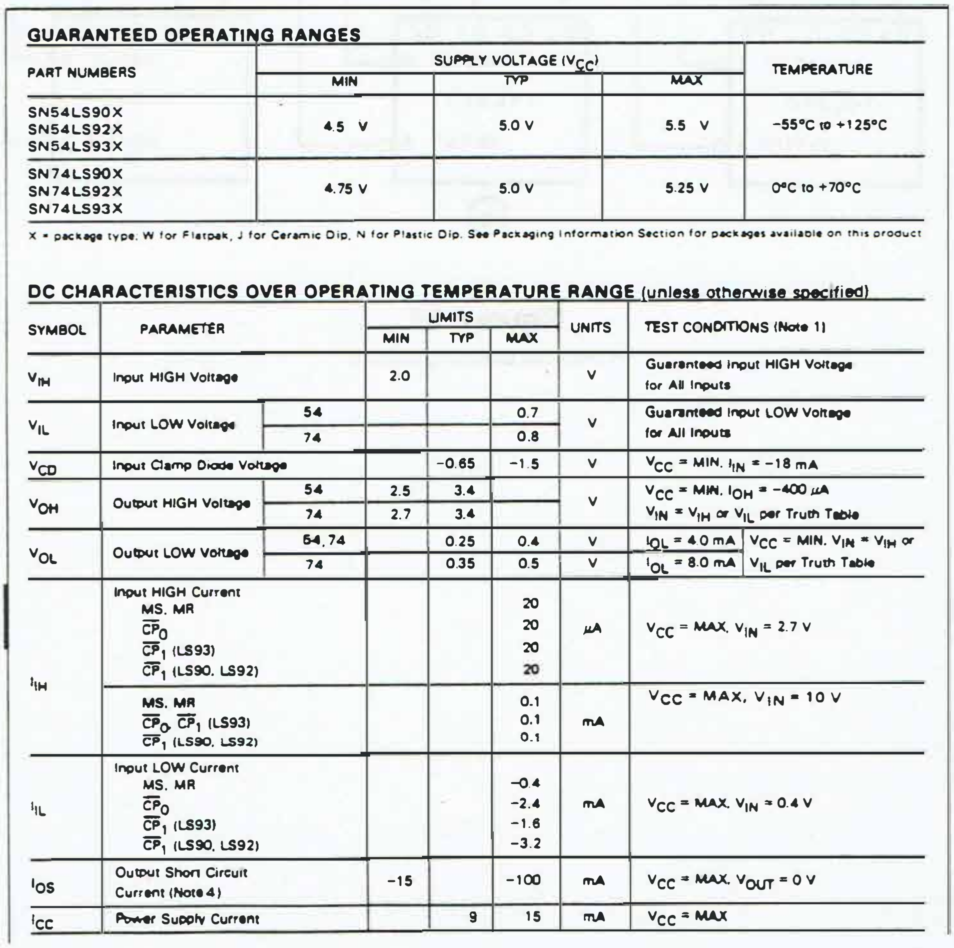
PART NUMBERS	SUPPLY VOLTAGE (V_{CC})			TEMPERATURE
	MIN	TYP	MAX	
SN54LS47X	4.5 V	5.0 V	5.5 V	−55°C to +125°C
SN74LS47X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type; W for Flatpak, J for Ceramic Dip, N for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (Unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Threshold Voltage for All Inputs
V_{IL}	Input LOW Voltage	54 74			0.7 0.8	V	Guaranteed Input LOW Threshold Voltage for All Inputs
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage, BI/RBO		2.4	4.2		V	V_{CC} = MIN, I_{OH} = −50 µA V_{IN} = V_{IH} or V_{IL} per Truth Table
V_{OL}	Output LOW Voltage BI/RBO	54, 74		0.25	0.4	V	I_{OL} = 1.6 mA — V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	I_{OL} = 3.2 mA
$I_{O(off)}$	Off State Output Current a thru g				250	µA	V_{CC} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table, $V_{O(off)}$ = 15 V
$V_{O(on)}$	On State Outout Voltage a thru g	54, 74		0.25	0.4	V	$I_{O(on)}$ = 12 mA — V_{CC} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	$I_{O(on)}$ = 24 mA
I_{IH}	Input HIGH Current				20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current BI/RBO Any Input except BI/RBO				−1.2 −0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS} BI/RBO	Output Short Circuit Current (Note 4)		−0.3		−2.0	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CC}	Power Supply Current			7	13	mA	V_{CC} = MAX

(b)



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE (V_{CC})			TEMPERATURE
	MIN	TYP	MAX	
SN54LS90X SN54LS92X SN54LS93X	4.5 V	5.0 V	5.5 V	−55°C to +125°C
SN74LS90X SN74LS92X SN74LS93X	4.75 V	5.0 V	5.25 V	0°C to +70°C

X = package type, W for Flatpak, J for Ceramic Dip, N for Plastic Dip. See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 µA V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA — V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA
I_{IH}	Input HIGH Current MS, MR $\overline{CP}_0$ $\overline{CP}_1$ (LS93) $\overline{CP}_1$ (LS90, LS92)				 20 20 20 20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
	MS, MR $\overline{CP}_0$, $\overline{CP}_1$ (LS93) $\overline{CP}_1$ (LS90, LS92)				0.1 0.1 0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current MS, MR $\overline{CP}_0$ $\overline{CP}_1$ (LS93) $\overline{CP}_1$ (LS90, LS92)				 −0.4 −2.4 −1.6 −3.2	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 4)		−15		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CC}	Power Supply Current			9	15	mA	V_{CC} = MAX

Figure 6-21

Sample data sheets for the 74LS47 (a) and 74LS90 (b).

The supply voltage ratings are found at the top of each respective sheet and the supply current at the bottom of each sheet. Notice that you have two choices for the supply voltage, depending on which temperature range the IC is designed to handle. I have used the ratings for the 0° to 70° devices. These devices require that the supply voltage be regulated between 4.75 and 5.25 volts. In other words, the supply voltage specification is 5 volts ± 0.25 volts. Most power supply modules specify regulation as a percentage of the typical output voltage. Since 0.25 volts is 5% of 5 volts, the power supply module must provide at least ± 5% regulation.

Next, you must determine the current requirements of the circuit. To do this, you should always use the maximum current ratings of the devices in the circuit. The three 74LS47s require a maximum current of 3 × 13 mA, or 39 milliamps, to operate. The three 74LS90s require a maximum current of 3 × 15 mA, or 45 milliamps. Finally, the maximum current required by each 7-segment LED is 7 × 20 mA, or 140 milliamps, when all seven segments are illuminated for maximum brightness. Consequently, the three displays will require a maximum of 3 × 140 mA, or 520 milliamps, to operate. Adding the above current requirements, you get:

$$\begin{aligned}\text{Max. Circuit Current} &= 39\text{ mA} + 45\text{ mA} + 520\text{ mA}\\ &= 604\text{ mA}\end{aligned}$$

As a rule of thumb, you should add at least a 25% cushion to the maximum current value. This takes care of unforseen circumstances due to Murphy's Law. In our example, the 25% cushion value is 0.25 × 604 mA, or 151 mA, bringing the total required supply current to 755 mA.

To summarize, the power supply for the counting circuit in Figure 6-20 must provide 5.0 volts, ± 5%, with a current capability of at least 755 mA.

Specifying the Power Supply Module

The final task is to specify the power supply module. Several typical modules are pictured in Figure 6-22. When you first begin to look at a manufacturer's catalog, you are overwhelmed with the variety of supplies that are available. This is illustrated by the typical supply listings in Figure 6-23. However, by comparing your circuit requirements to the supply specifications, you can quickly narrow the choices down to just a few candidates. For our example, the 5V supply module marked with an arrow in Figure 6-23 will do the job. Notice that this supply will provide 5 V, ± 0.25% with a current capability of 1 ampere. This is more than enough to meet our circuit requirements.

I should mention that a power supply's cost is directly proportional to its regulation qualities and current capabilities. If one can be found, a 0.75 ampere supply with less regulation will do the job for our counter circuit, resulting in a more economical design.

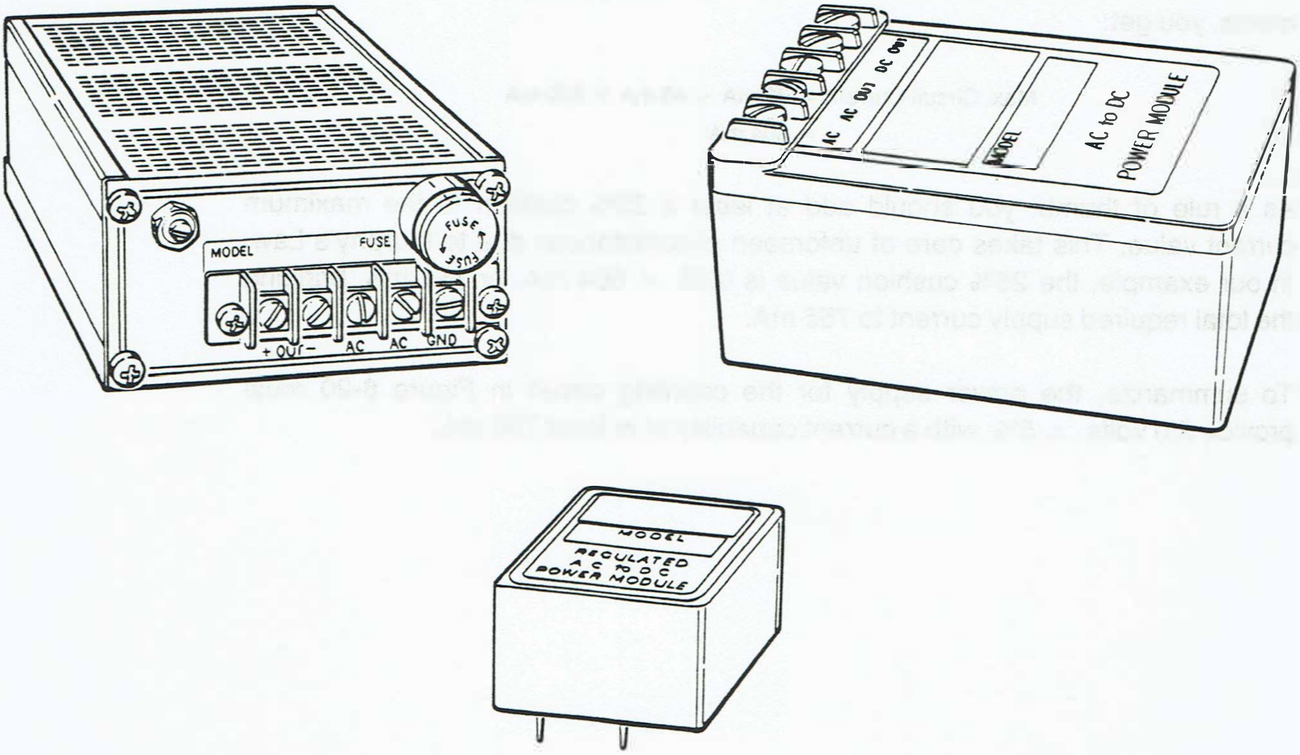


Figure 6-22
A few typical power supply modules.

Output Voltage	Output Current Amps.	Regulation
		Load ±%
1	.500	.4
1.5	.500	.3
1.5	1.0	.5
1.5	2.5	.6
2	.400	.25
3	.500	.25
3.6	.500	.15
3.6	1.0	.4
3.6	2.5	.4
4	.400	.15
5	.500	.15
5	1.0	.25
5	1.5	.35
5	2.0	.25
5	2.5	.25
6	.400	.1
6	.550	.25
6	1.0	.25
6	1.75	.2
7	.340	.1
7	.450	.2
7	.900	.25
7	1.15	.2
8	.300	.1
8	.700	.2
8	1.1	.2
9	.260	.1
9	.450	.15
9	.850	.2
9	1.5	.2
10	.240	.05
10	.400	.15
10	.750	.2
10	1.2	.15
11	.220	.05
11	.350	.15
11	.600	.15
11	1.0	.15
12	.200	.05
12	.400	.1
12	.700	.15
12	1.2	.2
13	.200	.05
13	.350	.1
13	.600	.1
13	1.0	.15
14	.200	.05
14	.300	.1
14	.500	.1
14	1.0	.15
15	.200	.05
15	.400	.1
15	.600	.1
15	1.0	.15
16	.160	.05
16	.350	.1
16	.500	.1
16	.900	.15
17	.140	.05
17	.325	.1
17	.450	.1
17	.750	.15
18	.120	.05
18	.270	.1
18	.400	.1
18	.550	.1
19	.120	.05
19	.250	.1
19	.400	.1
19	.700	.1
20	.120	.05
20	.200	.1
20	.400	.1
20	.700	.1
21	.120	.05
21	.175	.1
21	.375	.1
21	.600	.1
22	.100	.05
22	.150	.1
22	.300	.1
22	.500	.1
23	.100	.05
23	.200	.1
23	.300	.1
23	.600	.1
24	.100	.05
24	.200	.1
24	.350	.1
24	.600	.1
25	.100	.05
25	.190	.1
25	.325	.1
25	.550	.1
26	.080	.05
26	.170	.1
26	.300	.1
26	.450	.1
27	.080	.05
27	.160	.1
27	.300	.1
27	.500	.1
28	.080	.05
28	.150	.1
28	.300	.1
28	.500	.1
30	.080	.02
32	.070	.02
34	.060	.02
35	.050	.02
36	.050	.02
38	.040	.02
40	.030	.02
40	.060	.02
42	.030	.02
44	.030	.02
45	.030	.02
48	.030	.02
48	.050	.02
50	.030	.02
50	.050	.02
55	.040	.02
60	.050	.02
65	.050	.02
70	.040	.02
75	.030	.02
185	.025	Unregulated
185	.050	Unregulated

Figure 6-23

Some typical power supply module specifications.

Self-Test Review

11. List the three types of power supplies used for digital circuits.

A. ______________________________
B. ______________________________
C. ______________________________

12. The four major functional parts of a linear supply are ____________, ____________, ____________, and ____________.

13. State the major difference between a linear and switching power supply.

____

____

14. What advantage does a switching supply have over a linear supply?

____

15. A 12-volt CMOS circuit requires 0.5 amperes to operate. A linear power supply is used to provide power to the circuit. How much power is lost due to regulation, if 25 volts is applied to the input of a series regulator?

____

16. A switching regulator generates a square waveform with a 60% duty cycle. Assuming a 20 kHz switching rate, how much time is the switching transistor on in every cycle? ____________

17. How much DC voltage is presented to the load if the regulator in question 16 provides a 25-volt on-time level? ____________

18. A 5-volt power supply specifies ± 2% regulation. What range of voltages would you expect to see from this supply? ____________

19. A standard TTL device requires the same supply voltage regulation as a low power Schottky device. However, the current requirements for standard TTL devices are higher than those for low power Schottky devices. The data sheets for a 7447 and 7490 list a maximum I_{CC} level of 43 milliamps and 32 milliamps, respectively. What are the power requirements of the counting circuit in Figure 6-20 if standard TTL devices are used?

____

20. Which power supply module listed in Figure 6-23 should be used for the standard TTL circuit in problem 19? ____________

Answers

11A. Batteries

11B. Linear supplies

11C. Switching supplies

12. transformer, rectifier, filter, regulator

13. The type of regulator circuit used.

14. Less power consumption.

15. 6.5 W

16. 30 μS

17. 15 V

18. 4.9 to 5.1 V

19. 5V, ± 5%, at 931 mA

20. 5 V, at 1 A

Following is the solution to Questions 15, 16, 17, 18, and 19.

15. The voltage drop across the series regulator is:

$$25\text{ V} - 12\text{ V} = 13\text{ V}$$

If the load draws 0.5 amperes, the power loss due to regulation is:

$$13\text{ V} \times 0.5\text{ A} = 6.5\text{ W}$$

16. Assuming a 20 kHz switching rate:

$$\text{PERIOD} = T = 1/20\text{ kHz} = 50\ \mu\text{s}$$

Solving for the on-time in the duty cycle equation gives:

$$\text{Duty Cycle} = (T_{ON}/T) \times 100\%$$

or,

$$T_{ON} = (\text{Duty Cycle} \times T)/100\% = (60\% \times 50\ \mu\text{s})/100\% = 30\ \mu\text{s}.$$

17. With a 25-volt on-time level:

$$\text{DC Output} = (V_{ON}\text{ level} \times \text{Duty Cycle})/100\% = (25\text{ V} \times 60\%)/100\% = 15\text{ V}$$

18. Since 2% of 5 volts is 0.1 volts, you would expect to see a voltage range of 5 volts ± 0.1 volts, or 4.9 to 5.1 volts.

19. The 7447s require a maximum current of 3 × 43 mA, or 129 milliamps. The 7490s require a maximum current of 3 × 32 mA, or 96 milliamps. The maximum current required by the LEDs is 3 × 7 × 20 mA, or 420 milliamps. Adding the above, you get:

$$\text{Max. Current} = 129\text{ mA} + 96\text{ mA} + 420\text{ mA} = 645\text{ mA}$$

Adding a 25% cushion brings the total supply current to:

$$\text{Total Current} = 645\text{ mA} + (.25 \times 645\text{ mA}) = 806\text{ mA}$$

Thus, the standard TTL counting circuit requires 5.0 volts, ± 5%, with a current capability of at least 806 mA.

DIGITAL CIRCUIT PROCESSING

Up to this point in the design process, the digital circuit design has been all on paper. The next "logical" step in the process is to construct a working prototype and prepare the design for volume production. The intent of this section is to acquaint you with several techniques that are currently used to implement your design.

Webster defines a **prototype** as "the first thing or being of its kind." Thus, the first circuit constructed from a paper (or computer) design is called a prototype circuit. Phototype circuits are used to check-out and evaluate the theoretical design.

The prototyping idea is similar to writing a computer program. In most cases, the first generation of the program doesn't run at all. It must be "debugged" using various editing techniques. The prototype serves the same purpose for you as the circuit designer: it allows you to debug the design so that it will perform as intended, and meet all the application requirements.

In general, prototyping can be accomplished using either **hardware** or **software** techniques. Hardware prototyping techniques include **breadboarding, wire-wrapping**, and simple **PC board** construction. The prototype is constructed and evaluated before implementing the circuit in its final production form.

Software prototyping, on the other hand, makes use of **computer aided design**, or **CAD**, to simulate and debug the design with computer software. The idea being that the design can be implemented in its final form directly from the computer, without any intermediate hardware prototype circuit.

Hardware Prototyping

There are several ways to construct a hardware prototype of your design, from very simple solderless breadboarding to more sophisticated PC board prototypes. The following description will acquaint you with most of these popular hardware prototyping techniques. However, even an excellent text such as this cannot teach you all the "tricks" required for good hardware prototyping. That is a "hands-on" learning process. In other words, there is no substitute for experience when it comes to good and efficient prototyping.

SOLDERLESS BREADBOARDS

To perform the experiments in this course, you must use a solderless breadboard like the one pictured in Figure 6-24. These breadboards are ideal for small experimental circuit designs. They are particularly useful during the early stages of prototyping when many changes are likely to occur in the design. Circuits can be wired very quickly and they are easily modified.

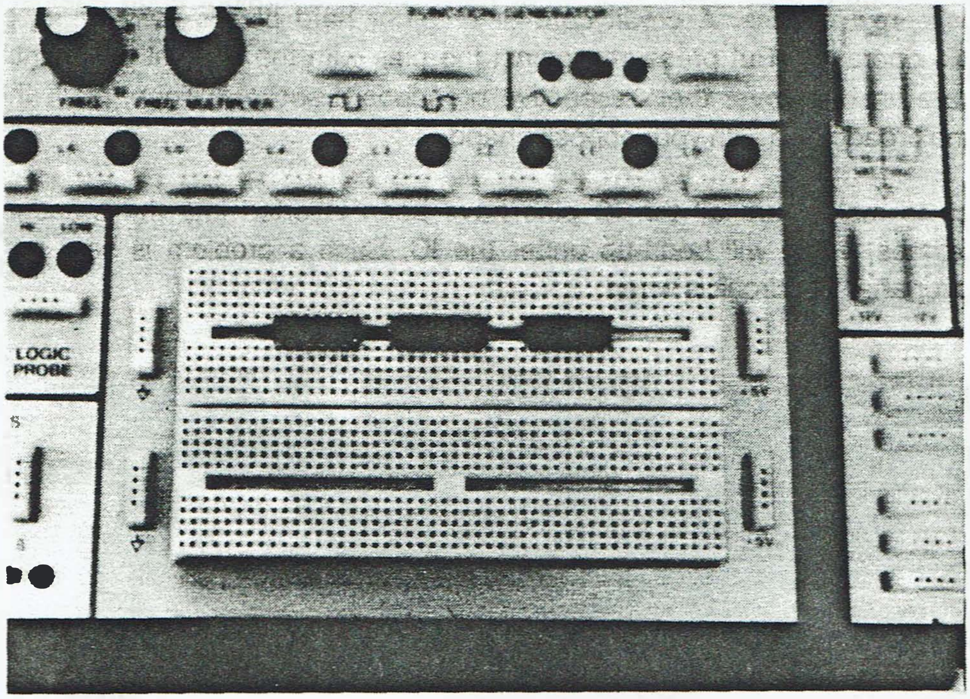


Figure 6-24
A typical solderless breadboard.
(*Photograph by Della Ann Benefiel.*)

As you can see from the figure, the ICs must stradle the breadboard slots. You must be careful when inserting the ICs, since many times the IC pins will not line-up with the breadboard sockets. This is especially true of new ICs. To insert an IC, I recommend that you position one row of pins over their respective breadboard sockets. Lay the pins in the sockets, but do not push them in. You will probably notice at this time that the opposite row of pins do not line-up with their breadboard sockets. If they do, go ahead and push the IC into the breadboard. If the pins don't line-up, hold the IC and tilt it slightly with one hand as shown in Figure 6-25. With your other hand, place a straight edge against the opposite row of pins. A credit card or or other hard plastic card works great for this operation. Put pressure against the pins with the straight edge until the pins are lined-up over their respective breadboard sockets. Then, push the IC into the breadboard with your opposite hand.

Check to make sure all the IC pins have been inserted into the breadboard. Sometimes, a pin will bend-up under the IC. Such a problem is very hard to discover after the circuit is wired.

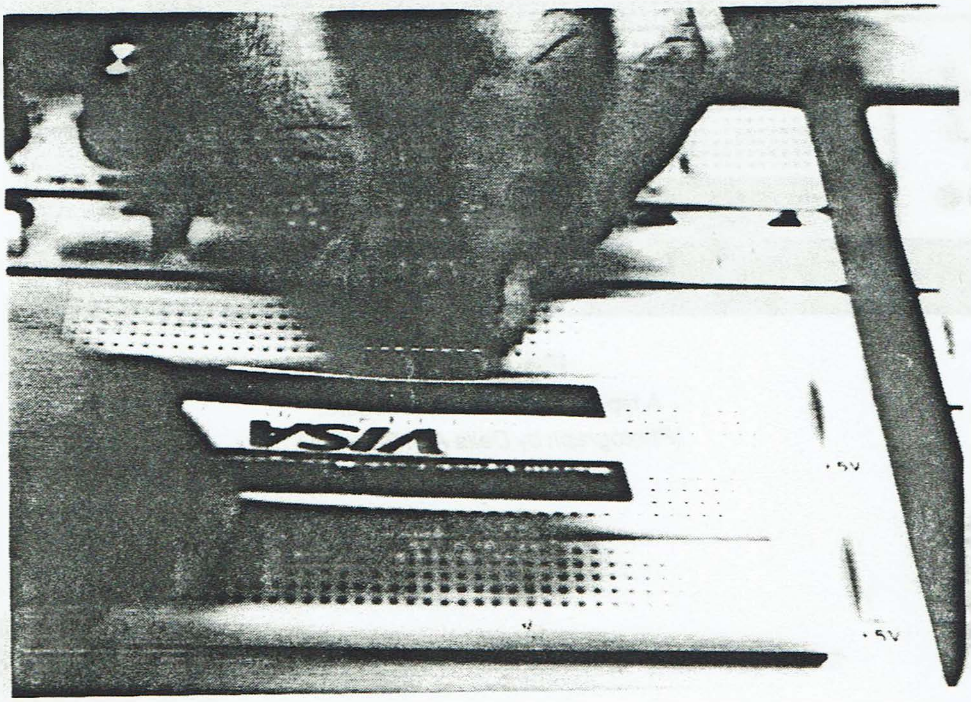


Figure 6-25
Proper IC insertion.
(*Photograph by Della Ann Benefiel.*)

Removing the ICs is just as important as insertion to prevent pin damage. **DON'T PULL THE IC'S OUT OF THE BREADBOARD WITH YOUR FINGERS**. This might seem like the fast way to do things, but it **always** results in bent pins. I suggest that you use a simple IC removal tool. An ordinary BiC® pen-top does an excellent job and is usually readily available. Push the pen-top into the breadboard slot, under the IC as pictured in Figure 6-26. In most cases, the IC will pop-up out of the breadboard as the pen-top is inserted. If not, rotate the pen-top from side-to-side and gently pry the IC out of the breadboard.

In most breadboards, the contact points in any given row are common. Notice that the contact rows run perpendicular to the IC slots. Other components, such as transistors, capacitors, and resistors can also be inserted into the breadboard, but lead thickness should not exceed 0.030 inch. In addition, most breadboards are designed to accept AWG #20 or #22 wire. Thicker component leads or larger wire will stress the internal spring contacts of the breadboard. When this happens, contact pressure is lost and electrical connections become unreliable.

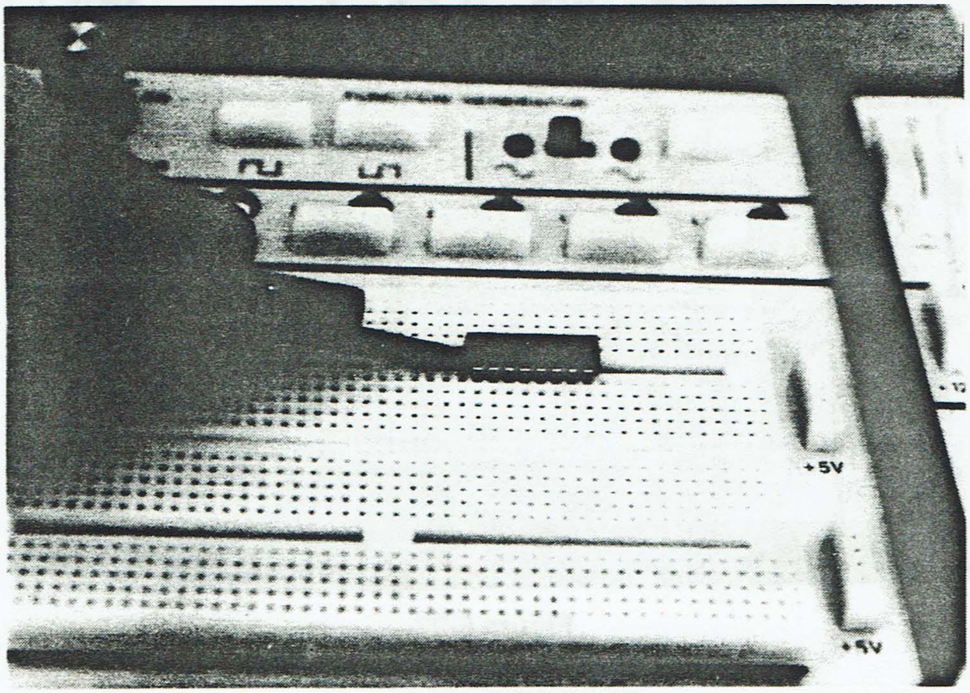


Figure 6-26

Proper IC removal.

(*Photograph by Della Ann Benefiel.*)

For a quick test of a simple circuit with only two or three ICs, you can wire the breadboard as shown in Figure 6-27. Here, precut jumper wires of random length have been used. Wires like this are usually the type found in a typical laboratory wire bin. Each wire arches up over the breadboard from one connection to another as shown. Although this is a relatively fast technique, it introduces several problems. First, the arched wires are difficult to trace with your eye when checking the circuit connections. Many times, you will need to wiggle a wire to find out where it goes. However, this often disturbs other wires, resulting in the accidental dislodging of several wire connections. Second, it is almost impossible to remove an IC and replace it without dislodging one or more wire connections. Finally, excess wire length may cause problems at higher clock frequencies, due to capacitive coupling of the clock signal between the wires. Capacitive coupling will cause the clock signal to actually bypass the ICs.

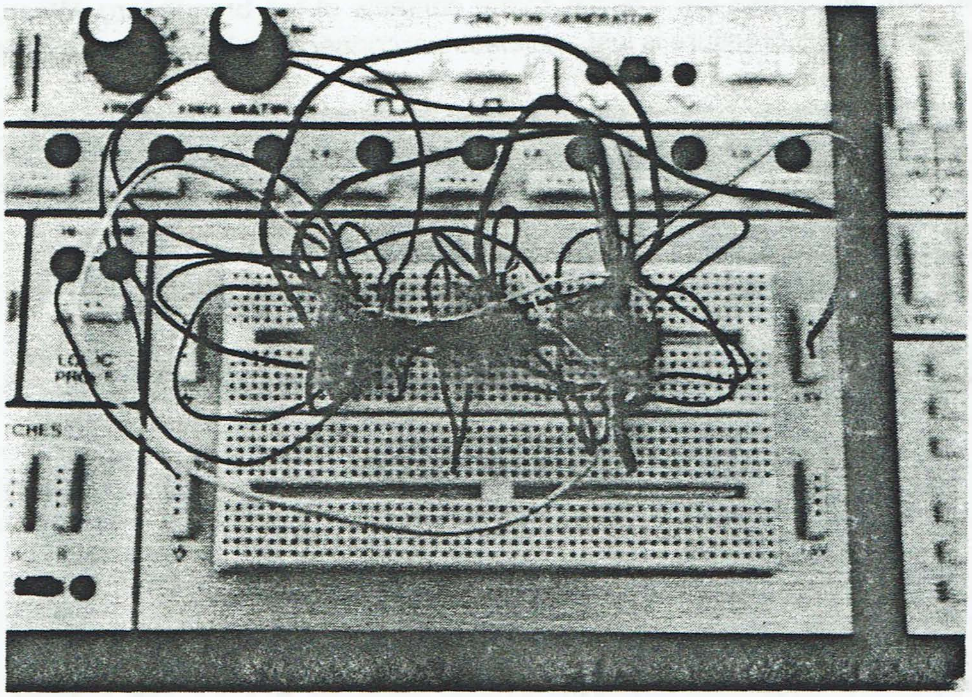


Figure 6-27

A quick but messy circuit. Should only be used for a quick test of simple circuits.

(*Photograph by Della Ann Benefiel.*)

A more reliable solderless breadboarding technique is to cut and strip each wire individually to the required length. This allows the wires to be routed along the breadboard in an orderly fashion as shown in Figure 6-28. As you can see, the wires are relatively short and can be easily traced. Furthermore, they do not cross over any ICs, thereby allowing easy IC replacement.

I recommend that you use AWG #22 solid copper wire with vinyl insulation. A sharp pair of diagonal (side) wire cutters, or an ordinary nail clipper, can be used to cut the wire very cleanly to length, without leaving a crimped or squashed end. This is important for easy wire insertion. A commercial wire stripper should then be used to strip about 3/8 inch of insulation from each end of the wire. A sharp thumbnail will also do the job if a commercial wire stripper is not available. With a little practice, you will be able to construct reliable breadboard circuits very quickly with this method.

Solderless breadboards should not be used for permanent or even semi-permanent installations. The internal contacts are not made with high quality material and they do not exert a very high contact pressure. Over a period of time, oxidation and other corrosive actions may cause a loss of the electrical connection at the contact points. In addition, vibration of the circuit can also result in loose connections.

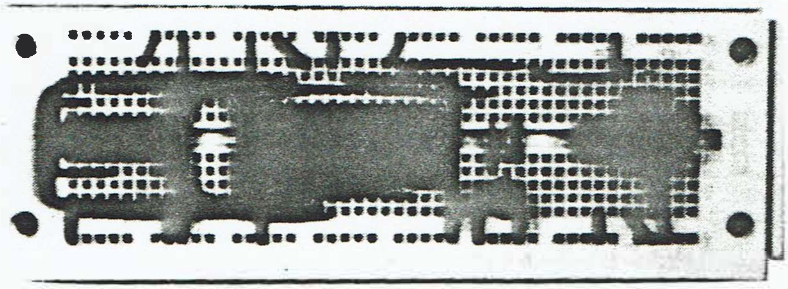


Figure 6-28

Recommended construction for solderless breadboard prototyping.

(*Photograph by Della Ann Benefiel.*)

PERFBOARD PROTOTYPES

A fairly common method of prototyping involves installing ICs and other components on perforated phenolic or epoxy boards as shown in Figure 6-29. Here, the component or IC socket leads are inserted through the perfboard. Wires are then soldered to the appropriate leads on the back side of the board. For lack of a better term, this type of circuit construction is often called **kludging**, implying a makeshift or messy arrangement. As you can see, the connecting wires criss-cross each other in a jumbled, "rat's nest" fashion.

Reliable circuit connections can be made by soldering wires directly to the protruding component leads. However, this creates the risk of component damage due to heat from the soldering operation. A better technique is to use IC sockets. Connections are then made by soldering wires to the protruding socket pins, rather than the component leads.

One of the principle features of the perfboard technique is its low material cost. Prepunched perfboard is available in various sizes and costs about 10 cents per square inch. The primary disadvantage of the perfboard method is that circuit construction is quite time consuming and tedious. In addition, modifications are difficult. You will often see the perfboard technique used when one or two components must be added, or patched, into an existing circuit.

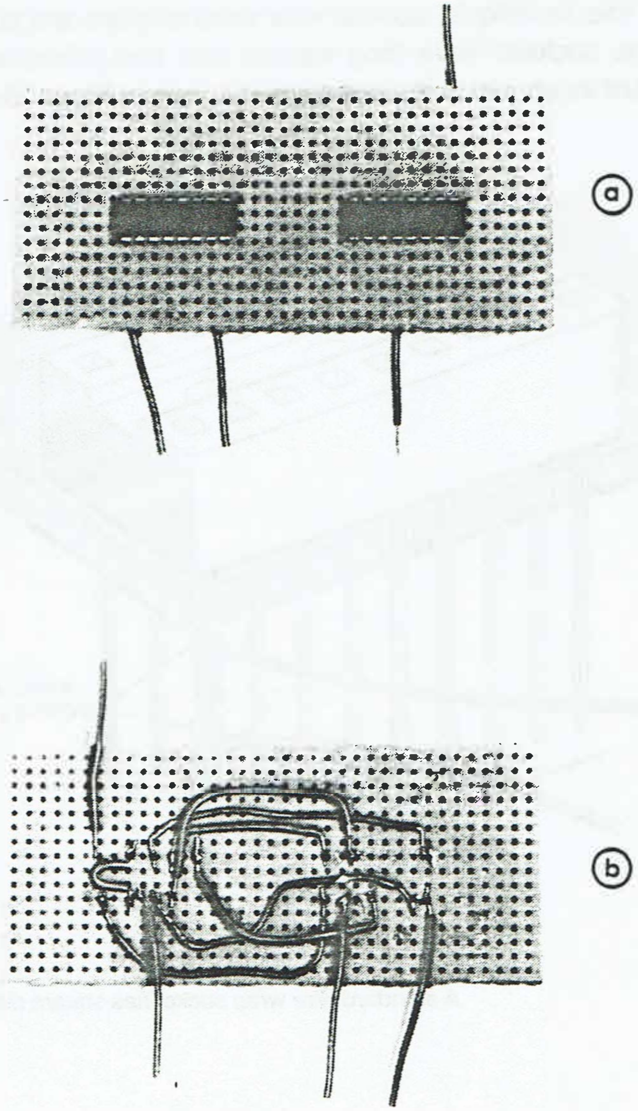


Figure 6-29
A perfboard prototype: top (a) and bottom (b).

WIRE WRAPPING

Wire wrapping is probably the most common technique used for circuit prototyping. With this technique, special wire wrap sockets are positioned on a wire wrap board. The sockets have long square pins that protrude through the back side of the board as shown in the cut-away diagram in Figure 6-30.

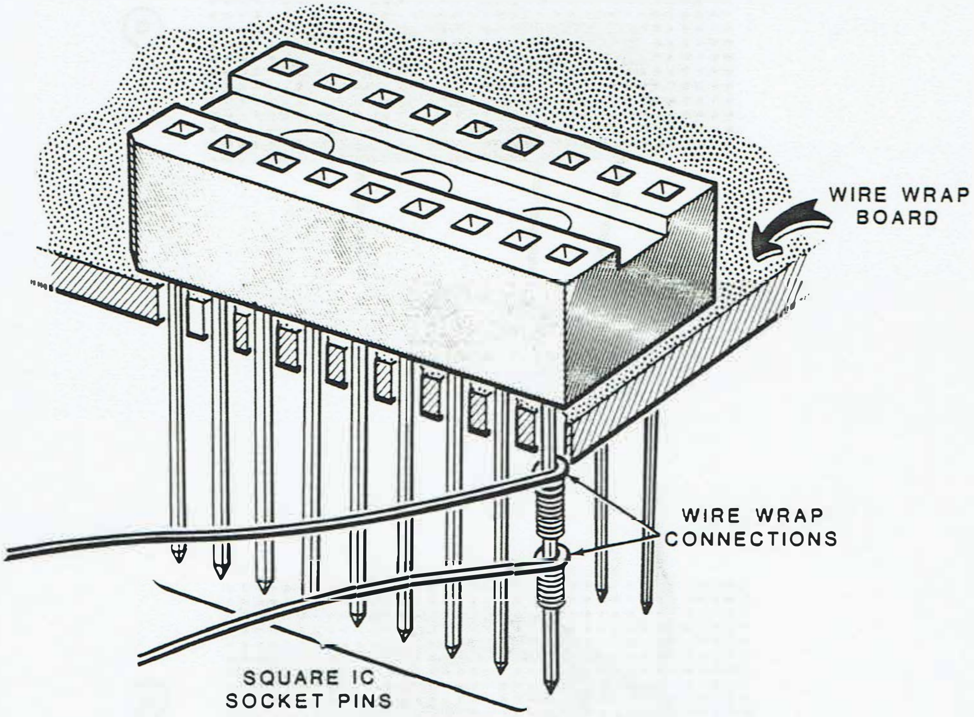


Figure 6-30
A standard wire wrap socket has square pins.

The wire wrap board can be simple perfboard with prepunched holes on a 0.1 inch matrix. This allows the sockets to be positioned in any rectangular pattern. Commercial wire wrap boards, like the one pictured in Figure 6-31, are also available. In most cases, these boards have pre-printed conductor patterns that allow common power, ground, and I/O connections. Observe that the board in Figure 6-31 has a pre-printed edge connector so that it can be plugged into a standard female card edge connector. Commercial boards range from simple 3 × 5 boards that handle three or four ICs, to high density boards that will accommodate over 100 ICs.

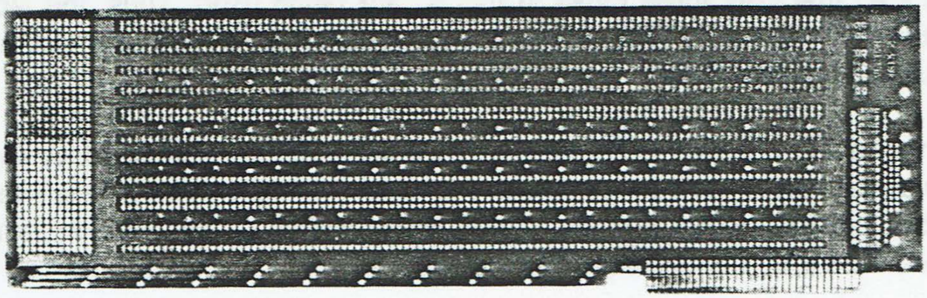


Figure 6-31
Typical commercially available wire wrap board.

Before beginning the wire wrapping process, an IC layout diagram should be made like the one shown in Figure 6-32. Here, nine ICs have been laid-out on a wire wrap board. The location of each IC is specified by the intersection of the respective drawing row and column. For example, the middle IC is located in section B2 of the diagram.

Each IC is labeled with its TTL or CMOS designation. Under this labeling, the schematic diagram location of each gate within the IC is specified. For example, the 74LS00 in the upper left-hand corner shows that its first NAND gate appears on page 1, section A3, of the schematic diagram. The second gate appears on page 2, section B5, and the third gate on page 1, section A2 of the schematic. Notice also that only three gates in this IC are being used. Thus, the labeling indicates that the fourth gate slot is not being used at present and is available for future circuit alterations.

Once the layout diagram is finalized, a **wire list** must be developed. A wire list shows all the wire wrap connections in the circuit. Since a given wire is wrapped from one pin to another pin, the list indicates which IC pin the wire is to be wrapped-from and where it is to be wrapped-to.

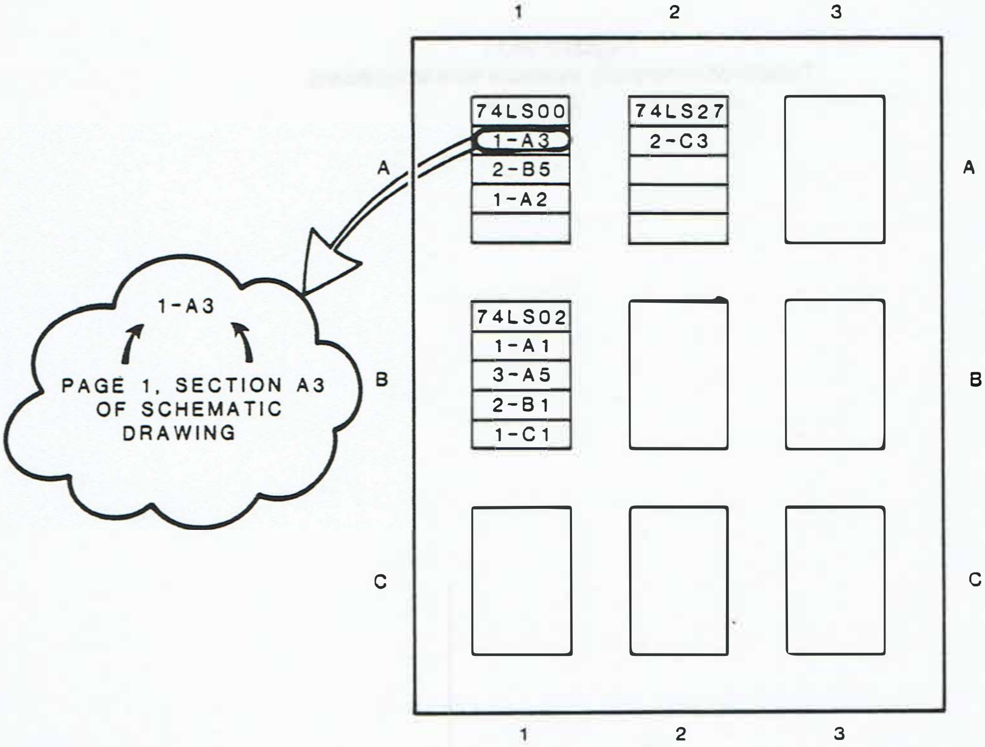
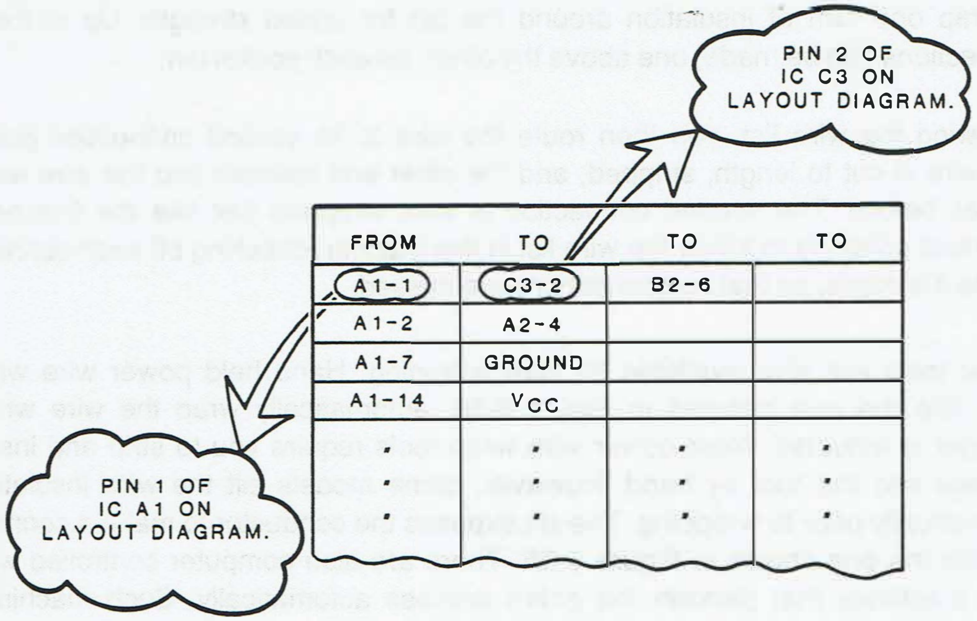


Figure 6-32
A typical IC layout diagram.

Look at the list in Figure 6-33. The first entry says that a wire must be connected FROM pin 1 of the IC located in section A1 of the layout diagram TO pin 2 of the IC located in section C3 of the layout diagram. Notice that A1-1 is the designation in the FROM column, and C3-2 is the designation in the first TO column. Likewise, the B2-6 designation in the second TO column means that a second wire must be connected FROM A1-1 TO B2-6.



FROM	TO	TO	TO
A1-1	C3-2	B2-6	
A1-2	A2-4		
A1-7	GROUND		
A1-14	V_{CC}		
.	.	.	
.	.	.	.
.	.	.	.

PIN 2 OF IC C3 ON LAYOUT DIAGRAM.

PIN 1 OF IC A1 ON LAYOUT DIAGRAM.

Figure 6-33
A typical wire list for the IC layout in Figure 6-32.

Once the wire list is completed, you are ready to begin wrapping the circuit. A special wire wrapping tool, like the one shown in Figure 6-34, is required to make connections between the socket pins. To make a connection, one inch of insulation must be stripped from a solid AWG #30 copper wire. The stripped end of the wire is inserted into the special wire wrapping tool, and the tool is placed over a socket pin and rotated. The tool must be rotated, while applying almost **no** downward pressure, until the entire one inch of exposed wire is wrapped around the pin. This usually requires about six turns. It is also a good idea to wrap one turn of insulation around the pin for added strength. Up to three connections can be made, one above the other, on each socket pin.

Following the wire list, you then route the wire to its second connection point. The wire is cut to length, stripped, and the other end inserted into the wire wrap tool as before. The second connection is then wrapped just like the first one. You must continue to follow the wire list in this manner, checking off each connection as it is made, so that no connections are missed.

Power tools are also available for wire wrapping. Hand held power wire wrap tools, like the one pictured in Figure 6-35, automatically wrap the wire when a trigger is actuated. Most power wire wrap tools require you to strip and insert the wire into the tool by hand. However, some models slit the wire insulation automatically prior to wrapping. The slit exposes the conductor to make a connection like the one shown in Figure 6-36. There are also computer controlled wire wrap machines that perform the entire process automatically. Such machines are used in commercial production processes and do not justify their expense for simple circuit prototyping.

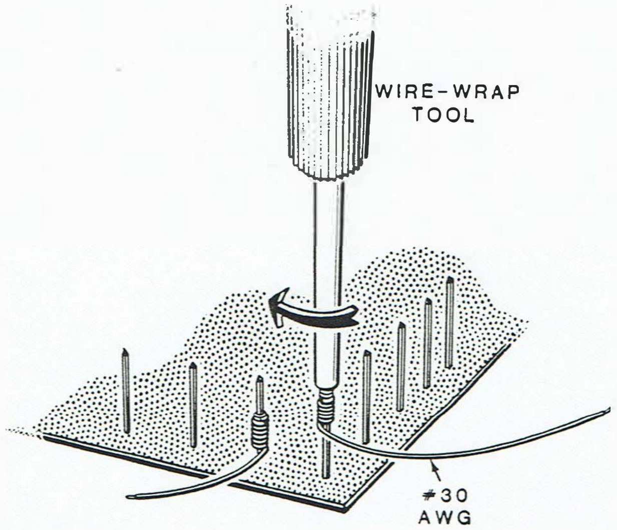


Figure 6-34
The wire wrapping operation.

Hand wire wrapping can be used very efficiently to construct even the most complex digital circuits. One of the key features of a wire wrap circuit is its design flexibility — modifications or additions can be made at any time. In addition, the reliability of a wire wrap circuit is excellent. High reliability is the result of the high pressure metal-to-metal contact created at each corner of the square wire wrap post, or pin. In fact, several large computer manufacturers, including IBM, have successfully used wire wrapped circuit boards for years in their large computers. For these reasons, I recommend wire wrapping for prototyping as well as low volume production runs.

Of course, wire wrapping does have its disadvantages. The special wire wrap sockets are more expensive than standard IC sockets. In addition, the long socket pins require extra spacing between the circuit boards and other system hardware. Lastly, wire wrapping more complex circuits by hand requires extreme patience to construct a mistake-free circuit. One simple mistake might result in many hours of troubleshooting.

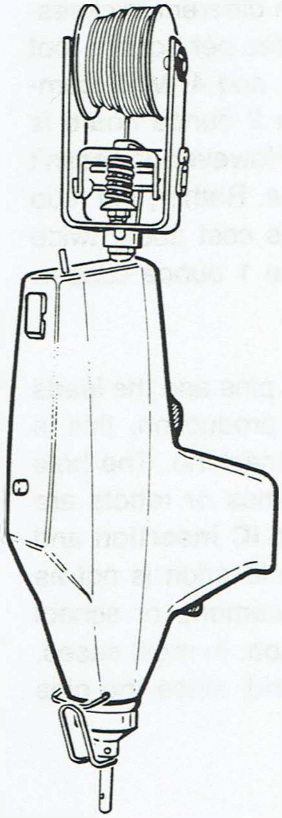


Figure 6-35
A power wire wrapping tool.

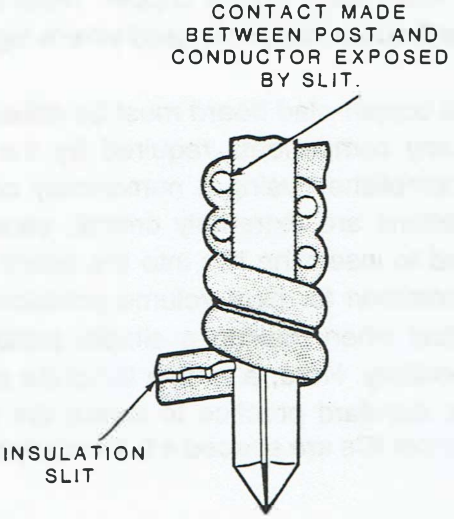


Figure 6-36
Some power wire wrap tools automatically slit the wire insulation prior to wrapping, to make a connection as shown.

PRINTED CIRCUIT BOARDS

Although printed circuit (PC) boards are sometimes used for prototyping, they are most often used for the final circuit implementation, after the design has been completely developed, evaluated, and debugged using other prototyping techniques. Nevertheless, you should be familiar with the different techniques that are used to construct PC boards, so that you can talk intelligently on the subject.

There are four fundamental steps that must take place during the construction of a PC board circuit. They are: **drilling, masking, etching**, and **soldering**. Of course, this assumes that the ICs have been laid-out and the associated PC board conductor pattern has been determined. This can be a major task in itself and often requires the aid of a computer. More about this later.

Drilling

The drilling process begins with a copper-clad board, which is nothing more than an epoxy or phenolic type board laminated on one or both sides with a thin layer of copper. These boards are commercially available with different thicknesses of copper. By convention, the copper is specified in ounces per square foot of board surface. There are five general grades: 0.5, 1, 2, 3, and 4. Most commonly used are the 1 and 2 oz/ft^2 grades. The copper on a 2 ounce board is nominally twice as thick as the copper on a 1 ounce board. However, it doesn't have twice the current carrying capacity, as you might imagine. Rather, the ratio is about 1.25 to 1. On the other hand, 2 ounce copper does cost about twice as much as 1 ounce copper. Most digital circuits only require 1 ounce copper. The 2 ounce copper is used where higher currents are involved.

The copper-clad board must be drilled to accommodate the IC pins and the leads of any components required by the circuit. In high volume production, this is accomplished using a numerically controlled, or NC, drilling machine. The hole locations are extremely critical, especially if automatic machines or robots are used to insert the ICs into the board. This is called **automatic IC insertion** and is common for high volume production runs. Of course, hole location is not as critical when making a simple prototype in your garage, basement, or school laboratory. Here, a simple template and drill press will do the job. In most cases, it is standard practice to layout the holes using a 0.1 inch grid, since the pins on most ICs are spaced a 0.1-inch apart.

Masking

The masking operation determines the copper conductor pattern of the board. Table 6-1 lists several methods that are commonly used.

Table 6-1: Some Common Masking Techniques

Technique	Comments
Pen Applicator	1:1 layout. One time only. Fast. Not totally reliable. Sloppy conductor pattern. Recommended for simple prototypes and circuit touch-up.
Tape Masking	1:1 layout. One time only. Slow and tedious. Recommended for simple prototypes and circuit patches.
Dry Transfer Masking	1:1 layout. One time only. Slow and tedious. Recommended for simple prototypes and circuit patches.
Photo Masking	1:1, 2:1, or 4:1 layout. One mask may be used over and over to make multiple boards. Extremely accurate. Recommended for all prototypes and short production runs.
Screen Print	1:1, 2:1, or 4:1 layout. One mask may be used over and over to make multiple boards. Good accuracy. Can be completely automated. Costly for low volume. Recommended for high volume production.

Pen Applicator — The easiest way to mask the conductor pattern onto the board is to simply draw it on with a special **resist pen**. The special pen ink protects the underlying copper from being etched away with acid during the etching process. After etching, all the copper areas under the ink remain to form the conductor pattern that you have drawn with the pen. In other words, the pen ink resists the acid etchant.

As you can imagine, the neatness of the resulting pattern is limited by your artistic abilities. In addition, only 1:1 layouts are possible, meaning that the size of the circuit you end-up with is the same as the mask pattern. This limits the conductor pattern detail, or resolution. Many times, the pen is used to touch-up other masking techniques, rather than generate the entire mask.

Tape Masking — The idea behind this method is similar to what you do with "masking tape" when you are painting. When painting, you cover any areas that are not to be painted with masking tape. With PC boards, you cover any copper that is not to be etched with **resist tape**. You can purchase special resist tape that adheres to the copper-clad board and resists the etchant during the etching process. Any copper areas covered by the tape remain after etching. The resist tape is available in a variety of pre-cut patterns as shown in Figure 6-37.

As you can imagine, taping a circuit is a slow tedious process. Usually, the hole pads are taped first, then the conductor paths are taped between the hold pads. A resist pen is often used to touch-up the taping, especially where a conductor path meets a hole pad. Only 1:1 layouts are possible, and only one circuit can be generated for each "tape-up."

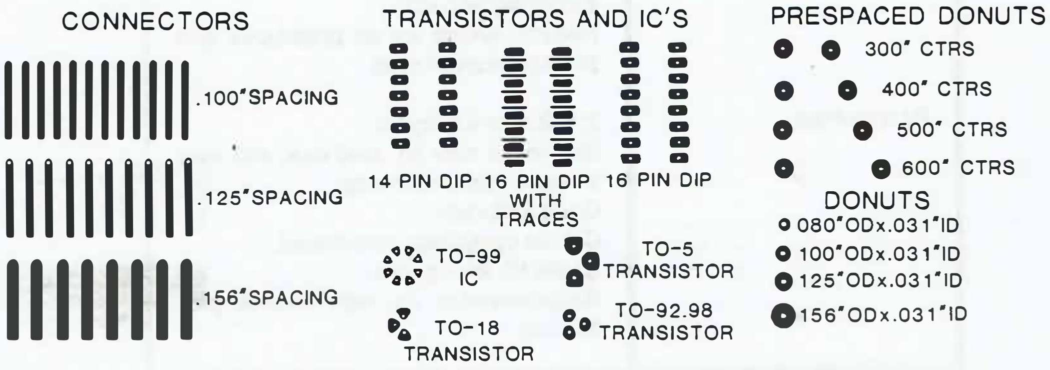


Figure 6-37
Typical tape mask or dry transfer artwork.

Dry Transfer Masking — This technique is similar to tape masking, except that the conductor lines, IC hole pads, and other artwork are on a sheet that must be rubbed to transfer the desired mask pattern to the copper-clad board. The transferred pattern protects the underlying copper by resisting the etchant solution. Again, this is a slow and tedious process. Only 1:1 layouts and one circuit per layout are possible.

Photo Masking — Photo masking is a more involved and costly process, but results in professional quality circuit boards. Once a photo-mask is produced, it can be used over and over to generate any number of boards. In addition, photo masking can be done on a 2:1 or 4:1 scale. This means that a mask can be produced that is two or four times larger than the resulting circuit. Then, using photo-reduction techniques, it is possible to make very accurate circuits with very high resolution.

The first step in the photo masking process is to produce a 1:1, 2:1, or 4:1 conductor pattern layout on a separate piece of mylar®. The 2:1 layout is most common. The layout is constructed using tape or dry transfer conductor lines, IC pads, and characters like the ones described earlier.

Once the layout is constructed, it must be photographed and reduced to a 1:1 pattern called a **photomask**. The photographing process generates a **negative** photomask. In other words, the conductor areas are clear and the non-conductive areas are black. A typical positive and negative photomask is shown in Figure 6-38.

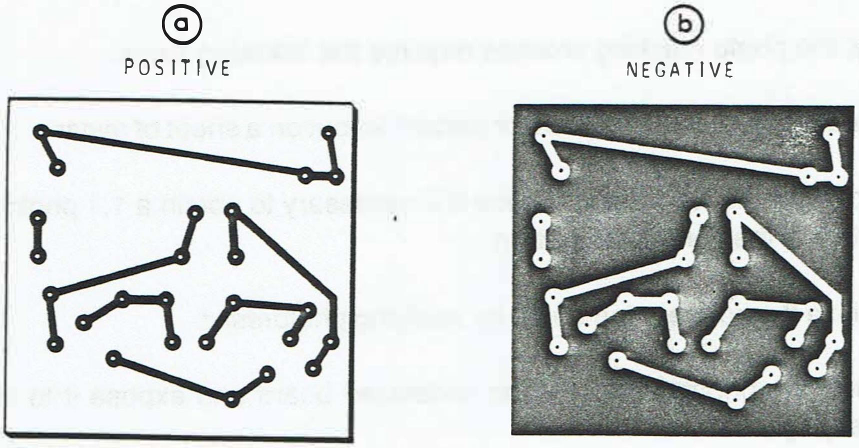


Figure 6-38

A typical positive (a) and negative (b) photomask.

The next step is to coat the copper-clad board with an emulsion called **photoresist.** Commercial photoresist is available as a liquid solution or in an aerosol spray can. When using the liquid solution, the board must be dipped in the photoresist. With the aerosol can, the board is sprayed. Since the photoresist is sensitive to light, it must be applied under darkroom conditions. A small 25 to 40 watt yellow "bug" lamp can be used instead of a red darkroom lamp. After the photoresist is applied, it must be cured in an oven for 10 to 15 minutes at about 115° F. If this all sounds too involved for you, you can purchase copper-clad boards that have already been "sensitized" with photoresist.

Once the board has been sensitized with photoresist, the next step is to position your photomask negative over the board and expose it to a high intensity light source. A **#2 photoflood** light source is common, however, ultraviolet light sources can also be used. In the absence of these, the old reliable sun will do the job — it just takes longer. Since the conductor areas of the photomask are clear, light will pass through these areas and expose the underlying photoresist. All non-conductor areas are not exposed to the light.

After the required exposure time, the photomask is removed and the board is developed, much like a picture. The developing solution disolves all unexposed, non-conductor, areas of the photoresist, and hardens the photoresist on all exposed, conductor, areas. You end-up with a board that has a hardened photoresist pattern that duplicates the desired conductor pattern. The hardened photoresist protects the underlying copper from being etched. However, the etchant will remove all unprotected copper in the non-conductor areas of the board.

In summary, the photo masking process requires the following steps:

1. Make a 1:1, 2:1, or 4:1 conductor pattern layout on a sheet of mylar.
2. Photograph the layout and reduce it if necessary to obtain a 1:1 photomask negative of the conductor pattern.
3. Sensitize the copper-clad board by applying photoresist.
4. Position the photomask over the sensitized board and expose it to a high intensity light source.
5. Develop the board to disolve the unexposed photoresist in the non-conductor areas.

You should be aware that two-sided boards require two photomasks, one for each side. As a result, two separate layouts are required, one on each side of **clear** mylar. The pattern that is to be used on the top, or **component side**, of the board is taped-up with red tape. The bottom, or **solder side**, of the board is taped-up with blue tape. Red and blue color filters are then used during the photographic process to generate two separate photomask negatives. Of course, a two sided board requires that both sides of the board be sensitized, exposed using each respective photomask, and developed. Naturally, you must be very careful to align the two photomasks so their holes match, or the circuit won't work. Figure 6-39 shows three typical methods for aligning the two photomasks.

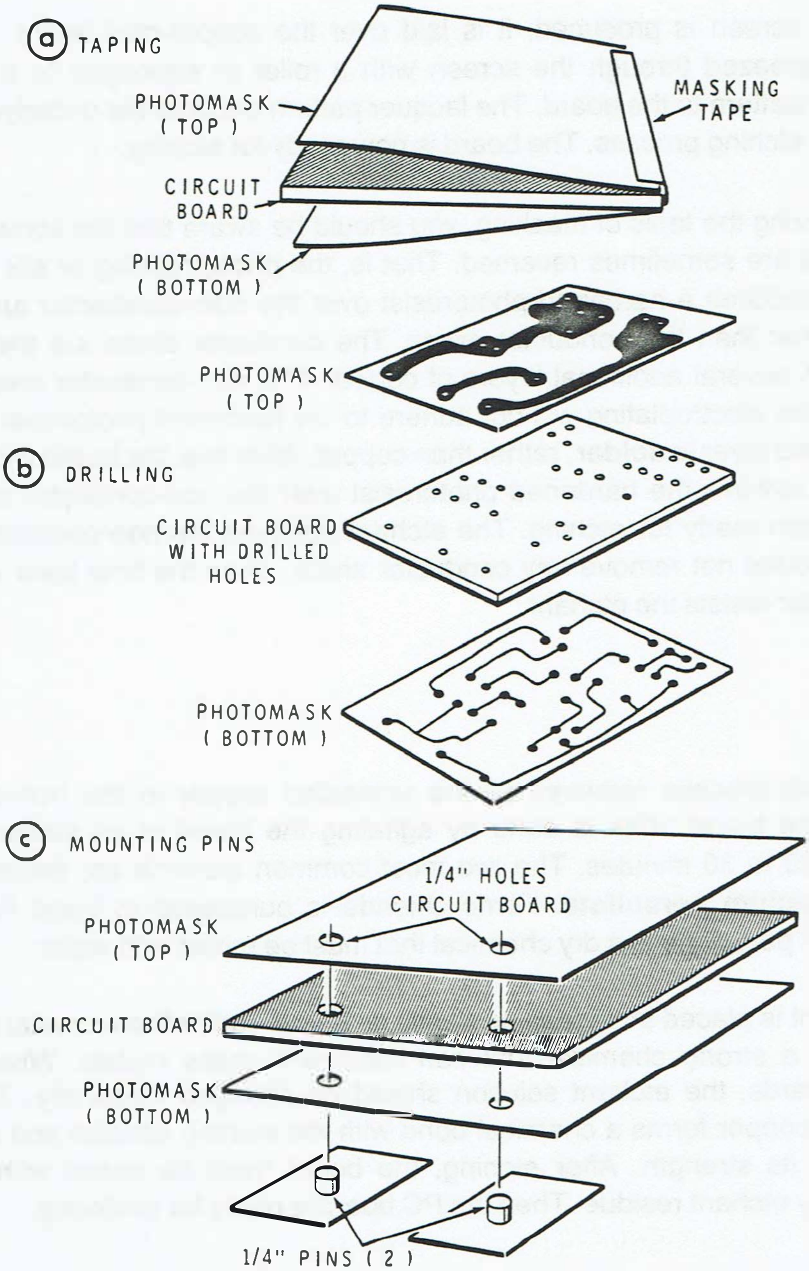


Figure 6-39
Photomask alignment for double-sided circuit boards.

Silk Screen Masking — The silk screening process also uses a photomask. However, this time a positive photomask is required. Thus, a mylar tape-up is constructed and photographed as before to produce a positive photomask whose conductor areas are black and non-conductor areas are transparent.

The next step is to coat a wire mesh silk screen with photoresist. The photomask is then placed over the coated screen and the assembly is exposed to ultraviolet light. The screen is developed and washed. This time, the process produces hardened photoresist in the non-conductor areas and an open screen in the conductor areas. Thus, the open areas of the screen show the final conductor pattern.

Once the screen is produced, it is laid over the copper-clad board. A lacquer is then squeezed through the screen with a roller or squeegee to transfer the conductor pattern to the board. The lacquer pattern protects the underlying copper during the etching process. The board is now ready for etching.

Before leaving the topic of masking, you should be aware that the above masking techniques are sometimes reversed. That is, the photomasking or silk screening process produces a hardened photoresist over the **non-conductor** areas of the board, rather than the conductor areas. The conductor areas are then electroplated with several additional layers of copper. The non-conductor areas are not plated, since electroplating will not adhere to the hardened photoresist. The final electroplated layer is **solder**, rather than copper. After this, the board is chemically treated to remove the hardened photoresist over the non-conductor areas. The board is then ready for etching. The etchant removes the non-conductor copper areas but does not remove any conductor areas, since the final layer of electroplated solder resists the etchant.

Etching

The etching process removes all the unwanted copper in the non-conductive areas of the board. This is done by agitating the board in an etchant solution for about 20 to 30 minutes. The two most common etchants are **ferric chloride** and **ammonium persulfate**. Ferric chloride is purchased in liquid form, while ammonium persulfate is a dry chemical that must be mixed with water.

The etchant is placed in a glass or plastic container rather than a metal container, since it is a strong chemical that can react with many metals. When etching several boards, the etchant solution should be changed frequently. This is because the copper forms a chemical bond with the etching solution and effectively decreases its strength. After etching, the board must be rinsed with water to remove any etchant residue. Then the PC board is ready for soldering.

I should mention at this time that double-sided PC boards often contain **plated thru-holes** to connect the conductors on one side of the board to those on the other side. These are holes in the circuit board that have their inner walls electroplated with copper. Thru-hole plating is a commercial process, requiring about 27 steps. As a result, it is not commonly seen on simple prototypes. However, you can purchase special **eyelets** in lieu of thru-hole plating for prototyping. The board is drilled where a conductor on one side must be connected to a conductor on the other. The eyelet is then inserted into the hole and crimped to make contact with both conductors.

Soldering

Prior to soldering, commercial boards are sometimes treated with an **electrodeless tint**. This is a chemical process that enhances the solderability of the board and helps prevent corrosion of the conductors. In addition, the copper conductors are sometimes electroplated with a tin-lead solder to increase their solderability. These are both optional steps which result in higher quality commercial boards.

After the board has been prepared, the components must be inserted into their respective holes and soldered. This is usually done by hand for simple prototypes, and by **wave soldering** for commercial production.

Hand soldering requires quite a bit of skill to make reliable circuit connections. If you have not had any experience with soldering, I suggest that you practice with scrap material to become familiar with the technique. You might also consider taking a course on soldering. As a general rule, use a low wattage (25 to 40 watts) soldering iron and 60/40 rosin core solder. DO NOT use acid core solder. It will damage the circuit board and it components.

Wave soldering is a commercial process that requires the board to be passed over a "wave" of solder. The solder wave is created by molten solder flowing out of a reservoir and down over an edge, similar to water flowing over a dam. The solder side of the board is then passed over the crest of the wave so that it barely contacts the liquid solder. As the board is passed over the wave, the molten solder adheres to all the exposed conductor areas of the board, completing the soldering operation in one motion.

Inspection and Quality Control Considerations

A PC board circuit must be inspected at several points during processing. The two most critical points are immediately after etching and soldering. After etching, the conductor pattern should be visually inspected for pits, voids, and bridges. These defects are illustrated in Figure 6-40. In addition, the proper conductor pattern should be verified in case a mistake was made during the masking process.

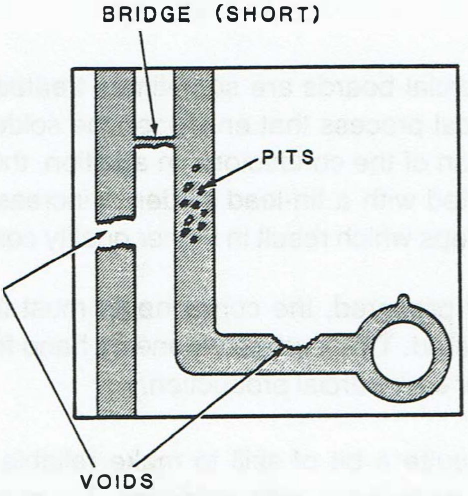


Figure 6-40
PC board conductor patterns must be inspected for pits, voids, and bridges.

For prototyping, visual inspection is usually performed by the individual technician or engineer building the prototype. In commercial production, visual inspection is often accomplished by comparing the conductor pattern to a reference pattern. This can be done manually using a reference pattern overlay and a comparison device called an **optical comparator**. The comparator magnifies the board and superimposes the reference overlay on the board. An inspector can then easily verify the proper conductor pattern and see any pits, voids, or bridges.

In very high volume production, the expense of a computer vision system is justified. The vision system performs the vision inspection task automatically by comparing the production boards to a reference board.

Aside from a visual inspection after etching, it is also a good idea to perform a continuity check on the conductor pattern. A standard volt-ohm-milliammeter, or VOM, can be used for this purpose. The VOM is discussed in the next unit, if you are not familiar with it.

Industry often uses a **bed of nails** to check the entire board for proper continuity. The bed of nails is actually a pin-contact layout for a given conductor pattern. The production board is placed on the bed-of-nails which is connected to computerized continuity tester. The tester then checks the continuity (resistance) of each conductor path.

The circuit board must also be inspected after soldering for proper soldering connections and correct component insertion. For prototyping, this can be done manually using a magnifying glass and a VOM. Look for any pits or voids that might indicate a bad solder connection. In addition, a VOM can be connected between the component lead on one side of the board and the solder connection on the other side to check for proper continuity. I suggest you place the VOM on a X100 resistance scale to avoid damage to the components. On this scale, a good connection should read almost zero ohms.

In addition to inspecting the solder connections, you should inspect to verify that the proper components have been inserted into the correct locations on the board. You wouldn't want a 7400 NAND gate and a 7402 NOR gate to be in each other's location. Furthermore, make sure the components are oriented correctly. It is very easy to insert an IC backwards. This often reverses the ground and supply potentials and, if not caught, will destroy the IC when power is applied to the circuit.

In industry, correct component insertion and orientation is performed by visually comparing the production board to a reference layout drawing which shows all the component locations and proper orientations. Sometimes, computerized vision systems are used to perform this task.

Software Prototyping

Software prototyping involves the use of **computer aided design**, or **CAD**, systems. Such a system can be a large mainframe or minicomputer system, or a smaller microcomputer system like the one shown in Figure 6-41. Of course, the mainframe or mini systems are very capable, but also very costly (over $100,000). On the other hand, a microcomputer system and associated software can be purchased for under $10,000 that will efficiently handle the prototyping task. In addition, many of the microcomputer systems use software programs that are upwardly transportable to a mainframe or minicomputer system.

A CAD system used for digital design should perform three basic tasks: **logic design, logic simulation,** and **PC board design.**

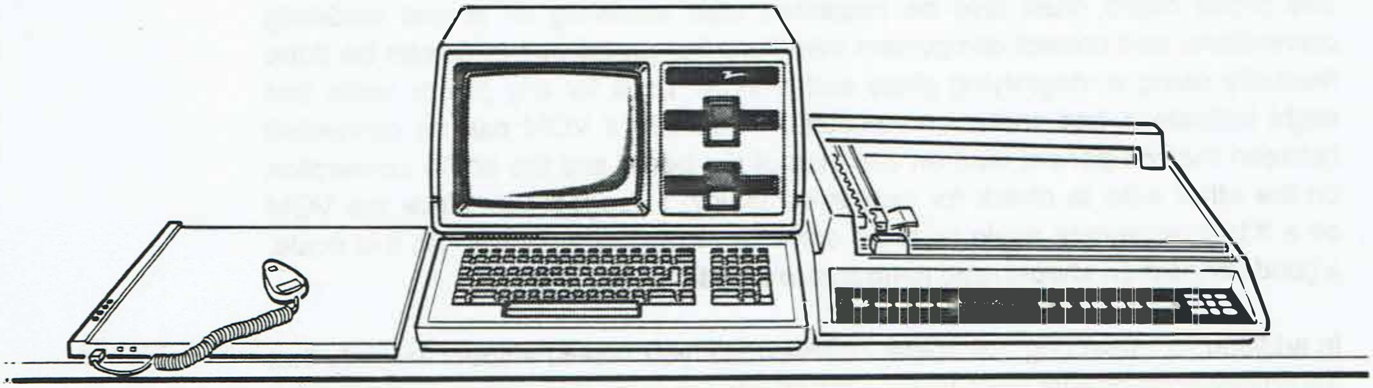


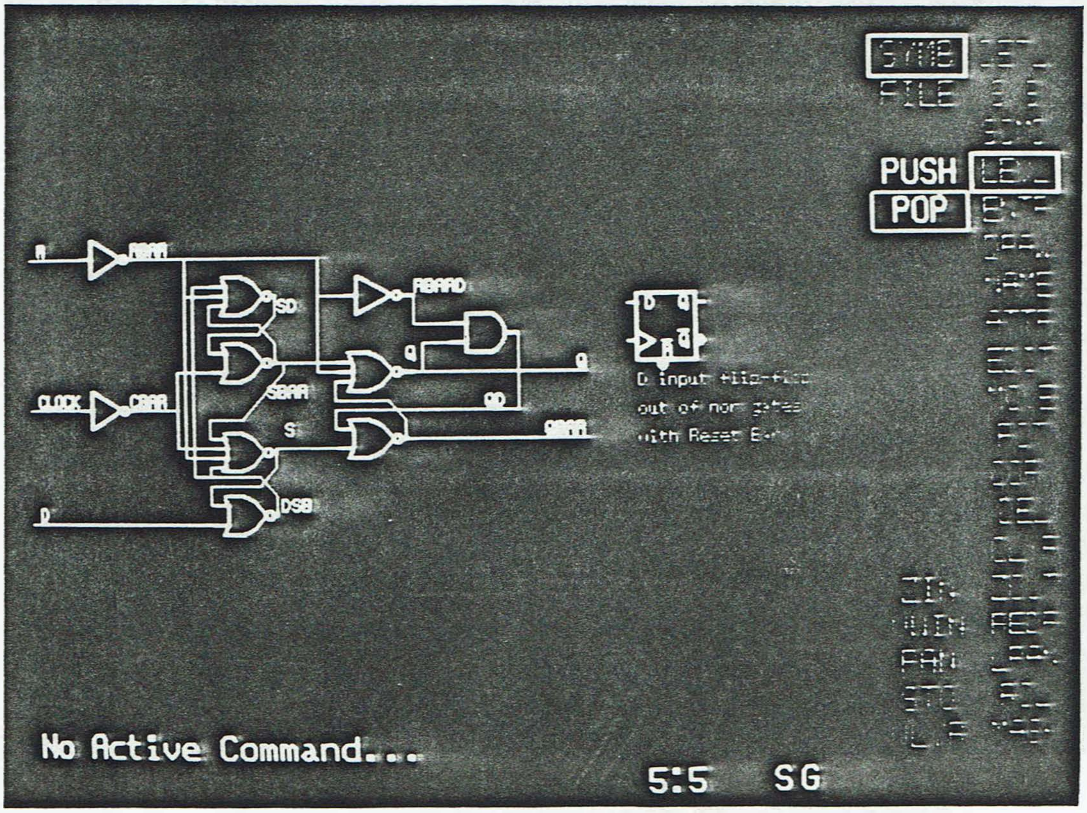
Figure 6-41
Typical microcomputer CAD system.

LOGIC DESIGN USING A CAD SYSTEM

When used for logic design, the CAD system aids you in designing the complete logic circuit from your black-box concept. You simply outline and label each functional black-box, showing the interconnections between them using the CAD system. The system then breaks each box down into smaller boxes, under your supervision, until the final gate level is reached. At the end of this "divide-and-conquer" process, you end-up with a completed logic design. In other words, the CAD system leads you straight through the digital design process, from top to bottom.

During the logic design process, the system generates displays such as the one shown in Figure 6-42. At this level, you insert, remove, and rearrange logic gates in order to meet a given black-box circuit function. When you are done, the system interconnects these smaller circuits per your original black-box definition and generates a completed logic diagram via a printer.

As computers get more intelligent due to developments in artificial intelligence, the CAD system will handle more of the design task. For instance, an intelligent system might generate a completed logic diagram for a time-of-day digital clock circuit by you simply labeling one of your black boxes as a "digital clock."



D input flip-flop
out of nor gates
with Reset B...

No Active Command....

Figure 6-42

A typical CAD display during the design phase.

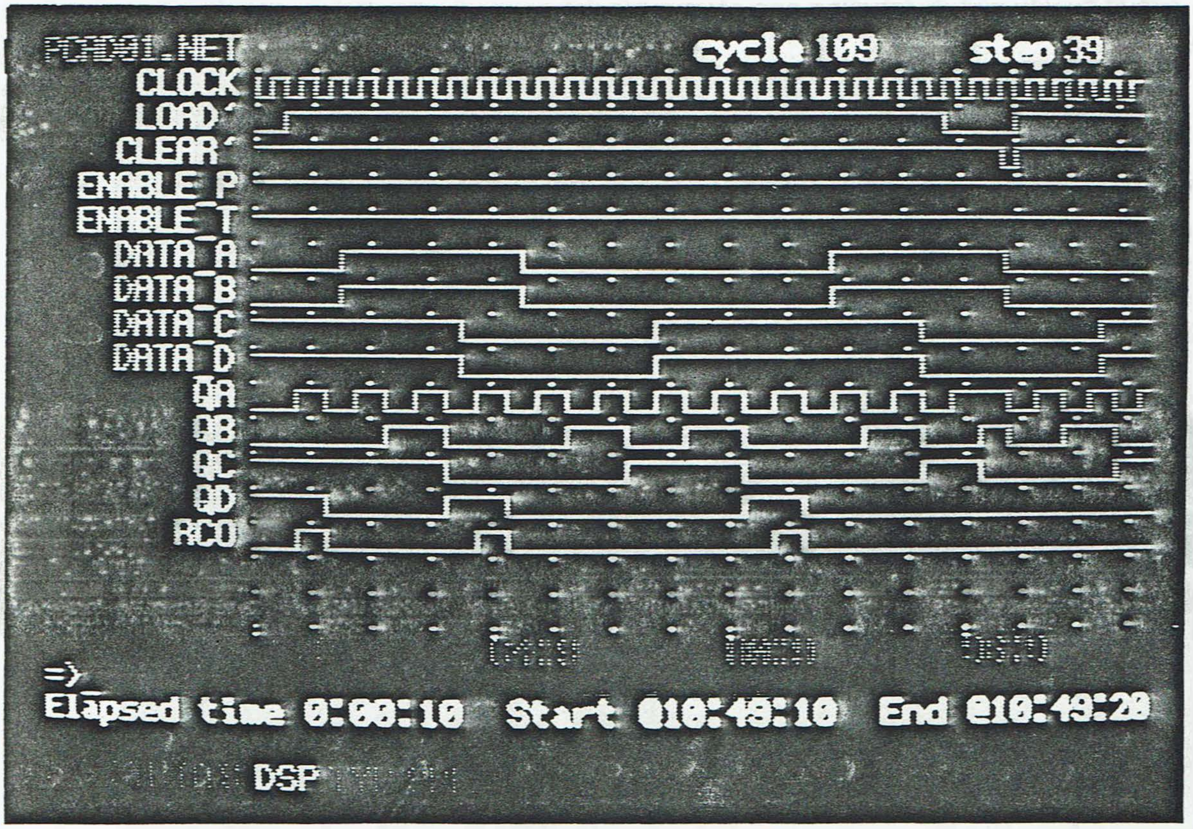
(*Photograph courtesy of Personal CAD Systems, Inc., Los Gatos, CA.*)

LOGIC SIMULATION, OR PROTOTYPING, USING A CAD SYSTEM

The idea behind software prototyping is that a computer is used to simulate the operation of the circuit. This takes the place of actually constructing and evaluating a hardware prototype. Consequently, after simulation, the design can be implemented on a large volume scale directly from your original design.

During logic simulation, the CAD system acts like a **software breadboard**. The logic design is stored in the system during the logic design phase. You must specify the input signals and the CAD system does the rest by simulating the operation of the hardware circuit. You then monitor the circuit outputs via timing diagram displays like the one shown in Figure 6-43.

You can observe the logic signals being generated in any part of the circuit at any point in time. In other words, you can use the system like a multi-channel oscilloscope to probe the design. You can check the output versus input operation of the entire design, or probe the logic of each circuit black box, down to the individual gates if necessary. The logic simulation can be frozen at any time and several outputs can be displayed simultaneously.



```
Elapsed time 0:00:10  Start @10:49:10  End @10:49:20
```

Figure 6–43

A typical timing diagram display during the simulation phase.

(*Photograph courtesy of Personal CAD Systems, Inc., Los Gatos, CA.*)

The real beauty of software prototyping is its flexibility. You are not limited by the ICs immediately available to you, as in hardware prototyping. In fact, there is no parts investment at all! In addition, you do not have to supply the required input logic and clock signals from logic switches, counters, or frequency generators. All of these signals are generated by software. Clock signals can be easily expanded, compressed, or frozen to isolate and trap timing problems. Its like having a complete digital laboratory literally at your finger tips.

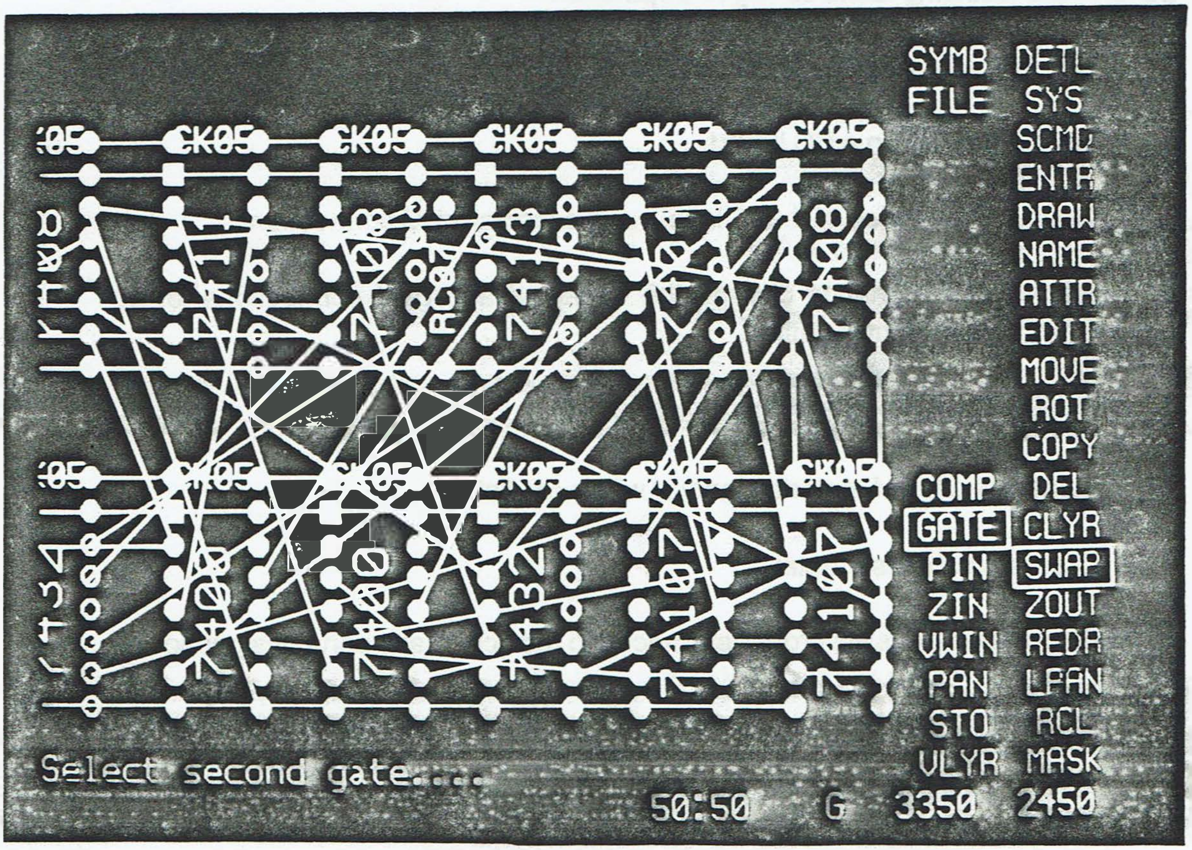
PC BOARD DESIGN USING A CAD SYSTEM

Once the CAD system has completed the logic design and simulation, it can be used to design the IC layout and PC board pattern. A typical system will perform the following tasks during this operation:

1. Compile all gates into physical IC devices.
2. Layout the ICs on a PC board.
3. Generate a display of the IC layout and associated IC connections.
4. Allow the designer to change the IC layout and gate connections for optimum circuit configuration.
5. Generate the PC board conductor pattern.
6. Generate a bill of materials and/or wire wrap list.
7. Draw a finished schematic.
8. Generate a parts list cross-referenced to the schematic parts designations.

As you can see from this list, the CAD system takes most of the work out of PC board design. First, the system automatically compiles all of the gates in your design into real IC devices. Then it performs a rough layout of these devices on a PC board and generates a display like the one shown in Figure 6-44. Notice that the IC layout is shown as well as a "rat's nest" connection diagram between the ICs.

At this point, you can change the rough layout by swapping or moving ICs. In fact, you can even swap gates within a given IC. While you are performing this editing function, the display is continually updated, showing you the result of any changes. Once the IC layout is finalized, the system can generate a wire wrap list if the circuit is to be wire wrapped. Otherwise, the system can be used to design the conductor pattern for a PC board.



Select second gate....

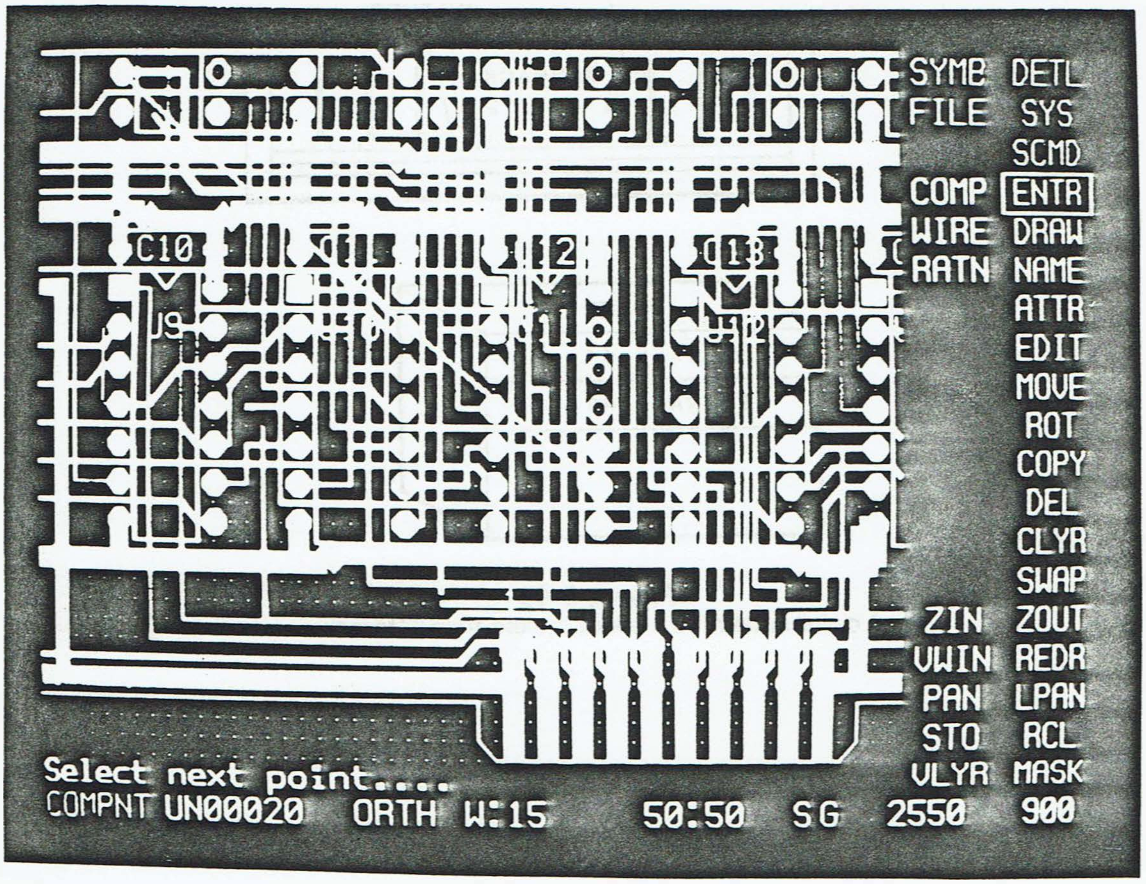
50:50 G 3350 2450

Figure 6-44
A typical IC layout and associated "rat's nest" display.
(*Photograph courtesy of Personal CAD Systems, Inc., Los Gatos, CA.*)

When a PC board is to be designed, the system will automatically generate and display a rough conductor pattern per your IC layout like the one shown in Figure 6-45. Again, you can edit the pattern and "stitch" jumper wires into the pattern where necessary. You can see the result of any changes immediately, since the system continually updates the display as the changes are made.

Finally, once the PC board design is finalized, the system will generate the associated engineering drawings, including a bill of materials. Drawings adequate for prototyping can be generated by an inexpensive dot-matrix printer or X-Y plotter. Production quality drawings for PC foil patterns, however, must be generated with a precision plotter. Quite often this type of plotter uses a high intensity light or a laser beam to accurately expose light sensitive film. That's all there is to the process!

In closing, I should mention that CAD systems, like the one discussed here, are a good investment if you are in the full-time business of designing digital circuits. Such a system would not justify its expense, however, if it is only being used on a hobby or part-time basis.



```
Select next point.....
COMPNT UN00020   ORTH W:15    50:50  SG  2550  900
```

Figure 6-45
A typical PC board conductor pattern display.
(*Photograph courtesy of Personal CAD Systems, Inc., Los Gatos, CA.*)

Shielding and Grounding

In Unit 1, you were acquainted with the modular approach to a digital system. This is illustrated in Figure 6-46. Recall that, once the system PC boards are constructed, they are inserted into a mother board which is part of a subsystem called a subrack, or chassis. Then, in larger systems, the subracks are placed in one or more racks. Whenever you connect digital circuit boards together to form a larger system, you must consider two important items: **shielding** and **grounding**.

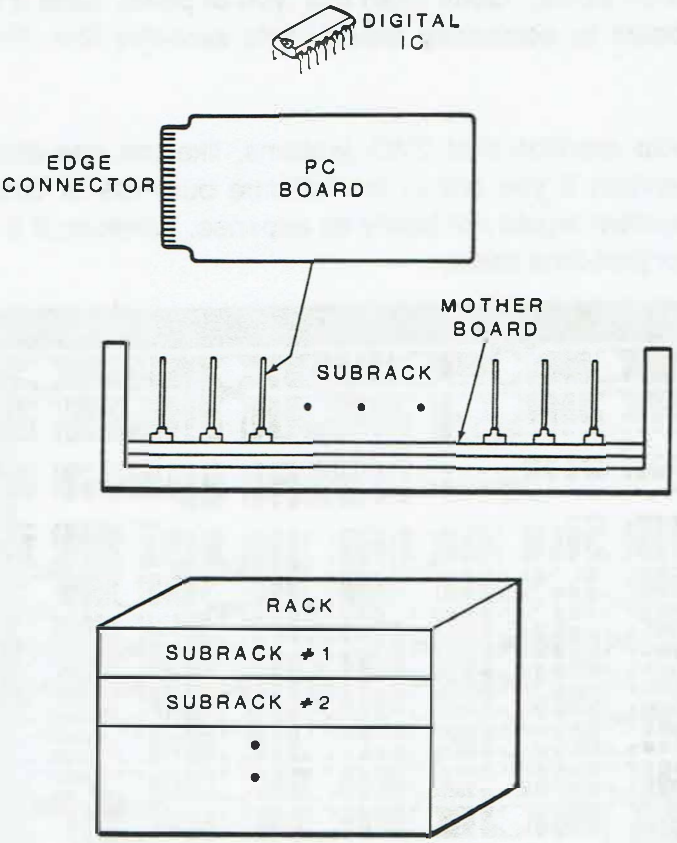


Figure 6-46
A digital system is a modular system, from IC to rack.

SHIELDING

Shielding is usually a consideration at the board level. As you are aware, an environmental factor that can affect the performance of digital devices is electromagnetic noise, commonly called **radio frequency interference**, or **RFI**. Although most digital ICs have a certain amount of built-in noise immunity, it is often not enough to prevent false triggering in noisy environments. Such environments not only include factories where heavy machinery is used, but also office and household environments. Think about it, even your house is filled with dozens of devices that generate electrical noise. Have you ever tried to watch TV when someone is running a food mixer or vacuum cleaner? How about listening to an AM radio when a TV is operating close by?

The most effective way to protect your circuit from electrical noise is to **shield** the circuit components as illustrated in Figure 6-47. Here, a metallic shield (cover) is placed on the circuit board, over the digital ICs. The shield is then connected to the ground used by the ICs within the circuit. This ground is called the **signal ground**, as opposed to the AC ground of the power supply line voltage.

Shielding prevents false triggering of the digital devices by blocking external electrical noise. In addition, the shield helps to prevent the high frequency oscillations of the digital circuit from being a noise source to other electrical circuits.

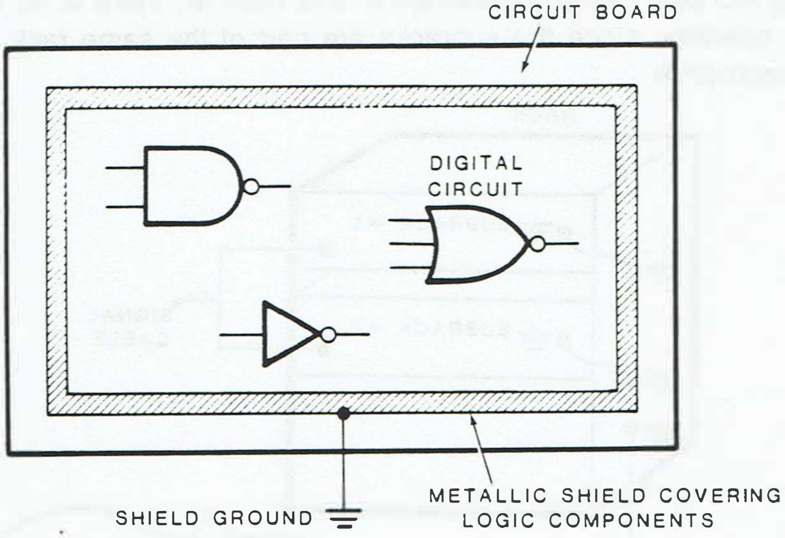


Figure 6-47
Shielding reduces noise problems.

GROUNDING

Many times, grounding is taken for granted during the design of a digital system. However, it can often be the source of improper circuit operation as well as potentially dangerous situations, for both the circuit and the operator. Grounding between PC boards within a given subrack is usually not a problem as long as a large ground plane is provided on the mother board. The ground plane is usually a wide printed conductor that provides a common ground path for all the PC boards plugged into the mother board.

Where grounding can become a problem is whenever one or more subracks, or **chassis**, must be connected together to form a larger system. There are two cases:

1. The subracks are placed in a common rack in close proximity to each other.
2. The subracks are separated by a relatively large distance.

The first case is illustrated by Figure 6-48. When the subracks are placed in a common rack, the signal ground in each subrack is connected to the rack. The rack is then grounded to the electrical ground of the AC system via the ground prong on the AC plug. The ground prong is the large round prong on a 3-prong AC plug. When connected in this manner, there is no danger to the circuit or operator, since the subracks are part of the same rack and relatively close to each other.

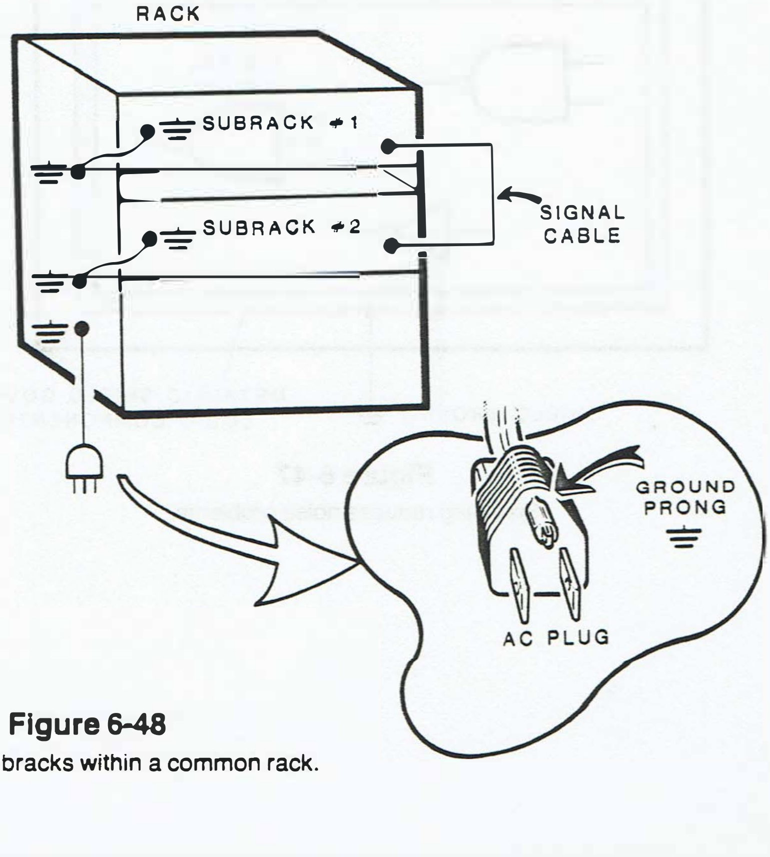


Figure 6-48
Grounding subracks within a common rack.

A grounding problem arises when two subracks, or chassis, must be separated by a large distance as shown in Figure 6-49. Here's why. The signals of each chassis are connected together via a common signal cable as shown. In addition, each chassis is connected to ground via its respective AC connector. Now, suppose the two AC circuits are controlled by separate circuit breakers. If this is the case, then the only common ground point is the main power service entrance to the building. Between each subrack and this point, other wires carrying AC current can induce a small voltage in each subrack ground leg. Most likely the voltage induced in one ground leg will be different from that induced in the other. This creates a difference in voltage potential between the two legs that results in a current flow within the ground loop due to Ohm's Law. Even the slightest potential difference in the two ground legs will create a relatively large, and potentially dangerous, current flow due to the small resistance of the grounding wire, since I = V/R.

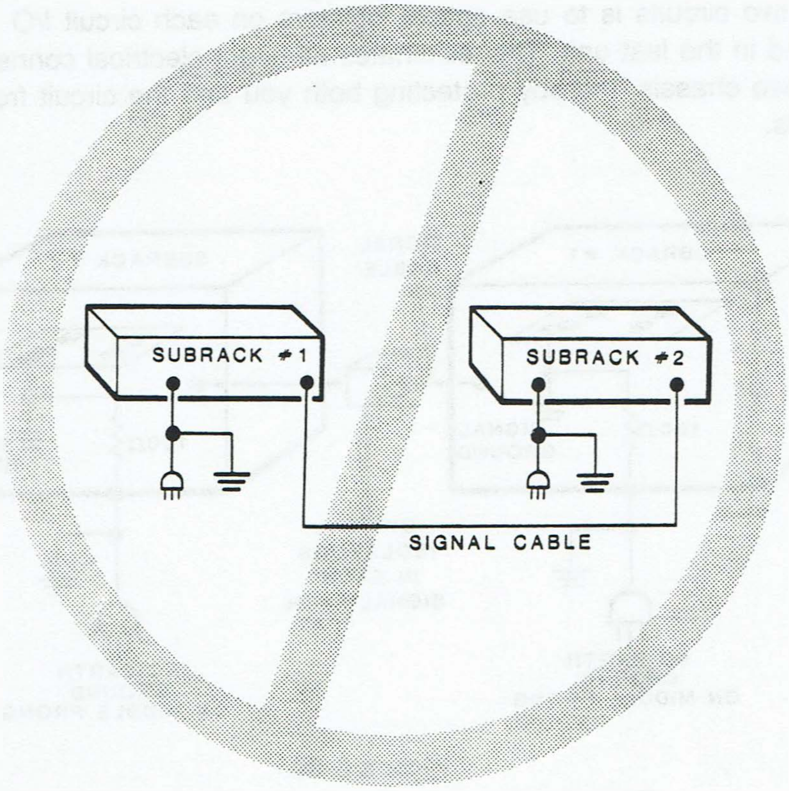


Figure 6-49

An unsafe, and potentially hazardous grounding situation.

The problem is even more dangerous if the two chassis are located in two different buildings. When this is the case, the only common ground point is good old mother earth. In this situation, the two building ground points can differ by several volts, creating even more current flow in the subrack ground legs. Even worse, if lightning strikes one of the buildings, a tremendous potential difference exists, resulting in equipment damage and a fire hazard. The point is this: it is **unsafe** to arbitrarily connect subracks together if they are separated by a relatively large distance, even across the room.

So, you ask: "How do I make it safe?" One way is to isolate the signal ground from the AC earth ground with a resistor as shown in Figure 6-50. Here, a 100 ohm resistor is inserted between each digital circuit ground and its respective chassis. Each chassis is then connected to earth ground via the AC plug ground prong. The effect of the resistor is to limit the amount of current that can flow in the ground loop, thereby reducing the danger. Of course, the **safest** way to isolate the two circuits is to use optical isolators on each circuit I/O signal line as discussed in the last unit. This eliminates all direct electrical connections between the two chassis, thereby protecting both you and the circuit from ground loop currents.

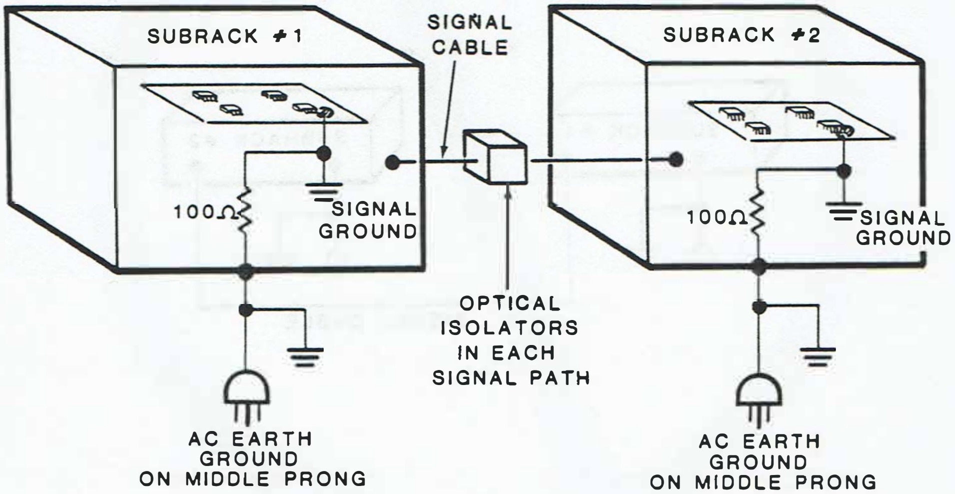

SUBRACK #1
SIGNAL CABLE
SUBRACK #2
100Ω
SIGNAL GROUND
100Ω
SIGNAL GROUND
OPTICAL ISOLATORS IN EACH SIGNAL PATH
AC EARTH GROUND ON MIDDLE PRONG
AC EARTH GROUND ON MIDDLE PRONG

Figure 6-50

Ground isolation resistors can be used to limit ground loop currents. However, the safest solution is to place an optical isolator in series with each signal path.

Self-Test Review

21. The two general methods for prototyping a circuit are __________ and ______________.

22. List at least three ways to construct a prototype circuit.

A. ______________________________

B. ______________________________

C. ______________________________

D. ______________________________

23. Name the two engineering documents that must be prepared before wire wrapping can begin.

A. ______________________________

B. ______________________________

24. A wire list indicates the following:

FROM	TO	TO
A2-5	B1-8	C2-3

A. How many wires must be wrapped? ______________

B. Where must the wires be connected? ______________

____

____

____

25. The four major steps in processing a PC board are:

A. ______________________________

B. ______________________________

C. ______________________________

D. ______________________________

26. Name at least three PC board masking techniques.

A. ______________________________

B. ______________________________

C. ______________________________

D. ______________________________

E. ______________________________

27. List the steps required to prepare a PC board for etching using a photomask.

A. ______________________________

B. ______________________________

C. ______________________________

D. ______________________________

E. ______________________________

F. ______________________________

28. The two most critical inspection points during the PC board process are:

A. ______________________________

B. ______________________________

29. A CAD system used for digital design is capable of performing what three design tasks?

A. ______________________________

B. ______________________________

C. ______________________________

30. When must special grounding or isolation techniques be employed for digital circuits? ______________________________

____

Answers

21A. Hardware

21B. Software

22A. Solderless Breadboards

22B. Perfboards

22C. Wire Wrapping

22D. Printed Circuit Boards

23A. IC layout diagram

23B. Wire list

24A. Two wires must be wrapped.

24B. One wire must be connected from pin 5 of the IC located in section A2 of the layout diagram to pin 8 of the IC located in section B1. The second wire must be connected from pin 5 of the IC located in section A2 to pin 3 of the IC located in section C2.

25A. Drilling

25B. Masking

25C. Etching

25D. Soldering

26A. Pen applicator

26B. Tape masking

26C. Dry transfer masking

26D. Silk screen

26E. Photo masking

27A. Make a conductor pattern layout on clear mylar.

27B. Photograph and reduce the layout to obtain a 1:1 negative.

Answers

27C. Apply photoresist to the copper-clad board.

27D. Expose the board through the photomask using a high intensity light source.

27E. Develop the board to dissolve the unexposed photoresist.

27F. Rinse the board with water.

28A. After etching

28B. After soldering

29A. Logic design

29B. Software prototyping through logic simulation

29C. PC board design

30. When two digital circuits, or subracks, are located some distance apart and not in the same rack.

SUMMARY

Digital circuits must often interface with the real analog world to sense and control events. This is especially true for non-computer applications of digital circuits. Such applications require conversions between digital and analog signals.

A device called a DAC is used for digital-to-analog conversions. DACs generate an analog voltage or current output signal that is proportional to the bit weights of a digital input signal. The size of a DAC is specified by the number of input bits that it can convert. The least significant input bit, or LSB, determines the output step size of the DAC. Each more significant input bit contributes twice the output level of the previous input bit. In this way, the DAC output is weighted by powers of two to produce an analog output that is proportional to the binary input. DACs are available to accept straight binary, BCD, two's complement, and offset binary inputs. Most DACs can also be wired to generate either a unipolar or bipolar output. Three important DAC specifications are resolution, accuracy (linearity), and settling time.

There are parallel output analog-to-digital converter devices, called ADC's, and serial output devices called V/F converters. The two most common parallel output ADC's are the successive approximation and integration type devices. The successive approximation ADC is the less costly and faster of the two, but the integration type is more immune to noise and more accurate. Both types have two control lines, called START and EOC, that are used to control the conversion process. An active signal applied to the start line starts a conversion. The ADC then generates an active transition on its EOC line when the conversion is complete.

V/F converters generate a series of TTL or CMOS pulses whose frequency is proportional to the analog input signal level. A counter circuit must be used to measure the V/F converter output. The count at the end of a given time period is proportional to the analog input. Three specifications that are important to all types of analog-to-digital converters are resolution, accuracy (linearity), and conversion time.

Nothing works without a power supply. Three types of supplies used for digital circuits are batteries, linear supplies, and switching supplies. The major difference between linear and switching supplies is the form of regulation employed. Switching supplies are much more efficient than linear supplies due to the use of switching regulators.

Power supplies are rated according to their output voltage level, % voltage regulation, and output current capabilities. In order to specify an adequate supply, you must first calculate the voltage level and regulation required by the circuit devices. Then, you must add-up the maximum current requirements of all the circuit devices. A 25% cushion is then added to account for unforseen conditions (Murphy's Law). The calculated voltage level, regulation, and current requirements are then used to specify an appropriate power supply module.

Once the circuit has been designed on paper, it must be prototyped for evaluation, prior to final implementation. There are two fundamental methods of prototyping: hardware and software prototyping. Hardware prototyping involves the actual construction of a prototype. You can choose from several hardware prototyping techniques, depending on the circuit complexity. These include solderless breadboarding, perfboard prototyping, wire wrapping, and PC board prototyping. Wire wrapping is by far the most common hardware prototyping technique.

Software prototyping is accomplished using a computer aided design, or CAD, system. The CAD system simulates the operation of the design using software, thereby eliminating the need to construct a prototype circuit. In addition, a CAD system can be used to design both the logic circuit and the PC board foil pattern required to implement the circuit. The final output of a CAD design is a complete set of engineering drawings that might include schematic diagrams, wire lists, component layout diagrams, PC board conductor patterns, and a bill of materials.

Lastly, the digital designer must consider how the circuit is to be shielded and grounded. In a noisy environment, shielding must be used to reduce the effect of radio frequency interference, or RFI. Special grounding precautions must be taken when two subracks, or chassis, are not located in the same rack. Direct connections between two digital circuits with separate ground legs can result in potentially hazardous ground loop currents that are dangerous to both the circuit and circuit operator.

UNIT EXAMINATION

1. If an 8-bit DAC has a 0-10 volt output range, the DAC step size is:
   - A. 1.25 V.
   - B. 0.39%.
   - C. 39 mV.
   - D. 0.039 mA.

2. The % resolution for the DAC in question 1 is:
   - A. 0.0039%.
   - B. 3.9%.
   - C. 39%.
   - D. 0.39%.

3. If the DAC in question 1 has an accuracy of ± 1/2 LSB, how close will the actual DAC output be to the expected output?
   - A. ± 0.078 volts of the expected value.
   - B. ± 39 millivolts of the expected value.
   - C. ± 19.5 millivolts of the expected value.
   - D. ± 0.0195 millivolts of the expected value.

4. Which of the following analog-to-digital converters does not have a parallel output?
   - A. Successive approximation.
   - B. Voltage-to-frequency.
   - C. Integration.
   - D. All of the above.

5. V/F converters are rated according to:
   - A. The number of output bits.
   - B. The number of analog inputs.
   - C. Their output frequency range.
   - D. Their input frequency range.

6. A 100 kHz V/F converter has a specified linearity of 0.01%. The output frequency of this converter will be within ________ Hz of the expected frequency.
   - A. 0.1 Hz.
   - B. 1 Hz.
   - C. 10 Hz.
   - D. 1000 Hz.

7. The two control lines of an ADC are usually labeled:
    - A. START and COE.
    - B. START and EOC.
    - C. RUN and Initiate Conversion.
    - D. BUSY and Data Valid.

8. The major difference between a linear and switching power supply is:
    - A. The rectifier circuit.
    - B. The transformer circuit.
    - C. The regulator circuit.
    - D. The filter circuit.

9. A digital circuit requires a +5-volt supply with ± 5% regulation at 0.75 amperes. What is the maximum power loss in a linear supply if 15 volts is supplied to the series regulator circuit?
    - A. 7.69 W.
    - B. 7.5 W.
    - C. 3.75 W.
    - D. 11.25 W.

10. A switching power supply produces a DC output of +12 volts. What is the maximum level of the switching regulator output if it generates a 50% duty cycle?
    - A. 24 V.
    - B. 0.24 V.
    - C. 6 V.
    - D. 0.06 V.

11. The power supply in question 10 specifies a ± 0.2% regulation. The maximum voltage you would expect to see from this supply is:
    - A. 14.4 V.
    - B. 12 V.
    - C. 12.24 V.
    - D. 12.024 V.

12. A power supply must provide an amount of current equal to the sum of all the typical current ratings of the individual circuit components.
    - A. True.
    - B. False.

13. Which of the following prototyping techniques require you to use AWG #30 wire?
    - A. Solderless breadboard prototyping.
    - B. Perfboard prototyping.
    - C. Wire wrapping.
    - D. PC board prototyping.

14. Which of the following prototyping techniques should **not** be used for permanent or even semi-permanent installations?
    - A. Solderless breadboarding.
    - B. Perfboard prototyping.
    - C. Wire wrapping.
    - D. PC board prototyping.

15. A wire list indicates the following:

FROM	TO	TO	TO
B2-3	A1-5	B2-4	A1-7

How many ICs are involved in the circuit connections?
    - A. One.
    - B. Two.
    - C. Three.
    - D. Four.

16. The photographic negative used while exposing a sensitized PC board is called a:
    - A. Photoresist screen.
    - B. Photomask.
    - C. Silk screen.
    - D. None of the above.

17. A PC board process that applies lacquer to the PC board prior to etching is:
    - A. Tape masking.
    - B. Dry transfer masking.
    - C. Photo masking.
    - D. Silk screening.

18. Which of the following tasks cannot be performed using a CAD system?
    A. Logic design.
    B. Prototyping.
    C. PC board design and layout.
    D. All of the above tasks can be performed using a CAD system.

19. The best way to reduce RFI is with shielding.
    A. True.
    B. False.

20. Potentially dangerous loop currents are caused by:
    A. Optical isolation.
    B. High resistance in the individual ground legs.
    C. Two ground legs at the same potential.
    D. Two ground legs at different potentials.

EXAMINATION ANSWERS

1. C — If an 8-bit DAC has a 0-10 volt output range, the DAC step size is 39 millivolts.

2. D — The % resolution for the DAC in question 1 is 0.39%.

3. C — If the DAC in question 1 has an accuracy of ± 1/2 LSB, the actual DAC output will be within ± 19.5 millivolts of the expected value.

4. B — A voltage-to-frequency converter does not have a parallel output.

5. C — V/F converters are rated according to their output frequency range.

6. C — If a 100 kHz V/F converter has a specified linearity of 0.01%, the output frequency of the converter will be within 0.01% of 100 kHz, or 10 Hz, of the expected frequency.

7. B — The two control lines of an ADC are usually labeled START and EOC.

8. C — The major difference between a linear and switching power supply is the regulator circuit.

9. A — A digital circuit requires a +5-volt supply with ± 5% regulation and 0.75 amperes. If 15 volts is supplied to the series regulator of a linear supply, the maximum power loss is:

$$(15\text{ V} - 4.75\text{ V}) \times 0.75\text{ A} = 7.69\text{ W}$$

10. A — A switching power supply produces a DC output of +12 volts. If the regulator generates an output with a 50% duty cycle, the maximum level of the switched output is:

$$(12\text{ V} \times 100\%) / 50\% = 24\text{ V}$$

11. D — If the power supply in question 10 specifies a ± 0.2% regulation, the maximum voltage you would expect to see from this supply is:

$$12\text{ V} + (0.002 \times 12\text{ V}) = 12.024\text{ V}.$$

12. B — False. A power supply must provide an amount of current at least equal to the sum of all the **maximum** current ratings of the individual circuit components. Ideally, you would add a 25% cushion to the power supply rating.

13. C — Wire wrapping requires you to use AWG #30 wire.

14. A — Solderless breadboarding should **not** be used for permanent or even semi-permanent installations.

15. B — A wire list indicates the following:

FROM	TO	TO	TO
B2-3	A1-5	B2-4	A1-7

Two ICs are involved in the circuit connections, since there are only two layout diagram locations specified: B2 and A1.

16. B — The photographic negative used while exposing a sensitized PC board is called a photomask.

17. D — The PC board process that applies lacquer to the board prior to etching is silk screening.

18. D — All the following tasks can be performed using a CAD system: logic design, prototyping, and PC board design and layout.

19. A — True. The best way to reduce RFI is with shielding.

20. D — Potentially dangerous loop currents are caused by two ground legs at different potentials.

UNIT 7

DEBUGGING THE DESIGN

CONTENTS

Introduction . . . . Page 7-3

Unit Objectives . . . . Page 7-4

Unit Activity Guide . . . . Page 7-5

Design Bugs . . . . Page 7-6

Test Instruments . . . . Page 7-22

Circuit Troubleshooting and Repair . . . . Page 7-37

Summary . . . . Page 7-54

Unit Examination . . . . Page 7-56

Examination Answers . . . . Page 7-60

INTRODUCTION

Now you have completed your design prototype, what would you say if I told you that there is less than a 50/50 chance that the circuit will work flawlessly the first time? Don't get discouraged, this is typical of new logic circuit designs, especially sequential circuits. This unit was written in an attempt to short circuit problems that might arise in your design. In addition, if problems do appear, the material in this unit will acquaint you with several test instruments and provide you with some general guidelines that can be used to find and correct your circuit ills.

UNIT OBJECTIVES

When you complete this unit, you should be able to:

1. List the four most common categories of combinational logic design bugs.
2. Prevent power supply glitches from affecting a digital IC.
3. Prevent problems due to floating inputs.
4. Name three potential timing problems in a sequential circuit.
5. Prevent timing problems due to unclocked inputs, clock skew, and excessive clock rates.
6. Calculate the maximum clock frequency for a sequential circuit.
7. Explain how to prevent hang-up states in sequential circuits.
8. Describe the features of the following test instruments and explain how they are used to troubleshoot digital circuits:
   A. Logic probe.
   B. Logic monitor.
   C. Logic pulser.
   D. Multimeter.
   E. Oscilloscope.
   F. Logic analyzer.
9. List seven general guidelines for servicing digital circuits.
10. List several forms of documentation that are important when servicing digital circuits.
11. Test the power supply and clock circuits for proper operation.
12. Troubleshoot and repair combinational and sequential logic circuits.

UNIT ACTIVITY GUIDE

☐ Read section on “Design Bugs.”

☐ Answer Self-Test Review Questions 1-10.

☐ Read section on “Test Instruments.”

☐ Answer Self-Review Questions 11-20.

☐ Read section on “Circuit Troubleshooting and Repair.”

☐ Answer Self-Test Review Questions 21-30.

☐ Study Summary.

☐ Complete Unit Examination.

☐ Check Examination Answers.

DESIGN BUGS

Some design bugs are obviously a result of improper or careless design practices. However, as the circuit becomes more complex, other bugs can creep into the design even though you have implemented the correct design procedures and have been very careful during the design process. This is especially true of timing problems in sequential circuits.

This section will uncover some of these invisible bugs and make you more aware of their presence so that you can anticipate problems and take corrective action before they occur. You will first examine design bugs common to combinational circuits, then those that are common to sequential circuits.

Combinational Logic Bugs

Design problems in combinational logic circuits usually fall into one of four categories: **noise, power supply glitches, floating inputs**, and **propagation delay**. You have already been acquainted with noise problems and how to shield your circuit to reduce the effect of noise. So, let's explore the remaining three problems and develop some simple preventive measures that can be applied to reduce these problems.

POWER SUPPLY GLITCHES

Logic gates require the most power when they are switching from one state to another. When several gates switch at one time, there is a sudden demand for current from the power supply due to an increased circuit load. Even though the supply is designed to handle the demand, its suddenness causes a high current spike, called a **glitch**, to appear on the power supply line before the regulator circuit can react and adjust to the increased load. Since the power supply is common to all devices in the circuit, all the gates "see" the glitch. Furthermore, since the logic levels of the circuit are established by the supply voltage, the glitch might cause temporary changes in logic levels, resulting in erratic circuit operation. This is especially true of TTL circuits because of their relatively narrow logic level bands.

The easiest and most effective way to prevent power supply glitches from affecting the circuit operation is to place a **bypass capacitor** across the +V and ground pins of each IC in the circuit. This is shown in Figure 7-1. For most SSI ICs, the capacitor is connected between pins 7 and 14. The bypass capacitor protects the IC by shunting, or bypassing, the glitch around the IC to ground. The capacitor value can be anywhere from 0.01 μF to 0.05 μF and should be mounted as close as possible to the respective IC.

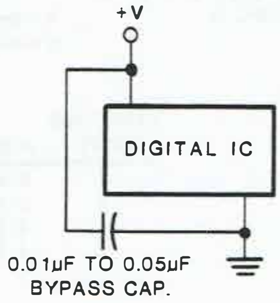


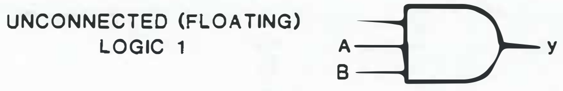
Figure 7-1

Bypass capacitors remove power supply glitches.

FLOATING INPUTS

In Unit 6, you found that shielding was the most effective way to prevent external noise from affecting the operation of a digital circuit. Even with shielding, noise generated within the circuit itself might affect certain IC inputs. These are the unused, or **floating**, inputs. Floating gate inputs are those that are not being driven by the output of another gate.

A simple example of a floating input is the 3-bit AND gate shown in Figure 7-2. Ideally, a TTL logic device input that is left unconnected will "float" up to a logic 1 state. Knowing this, you might suggest that to make a 2-bit AND gate from a 3-bit AND gate, you simply leave the extra input unconnected, or floating, as in Figure 7-2. Since the unconnected input floats to a logic 1 state, the device will always act like a 2-bit AND gate, right? Well, sometimes, but not always. Noise and power supply glitches can easily change the state of the floating input.

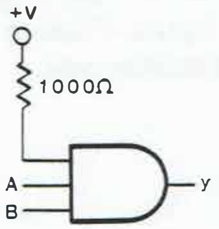


FLOATING INPUT	A	B	y
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Figure 7-2

A floating, or unconnected, TTL input "should" float up to a logic 1 level.

A simple rule to prevent problems due to floating inputs is to hardwire all unused inputs to the +V supply via a 1000 ohm pull-up resistor as shown in Figure 7-3. This holds the input high and does not allow it to arbitrarily change state. In addition, if the input must ever be used, another gate output can be connected to the input without removing the pull-up resistor. Why? Of course, you can also tie an unused TTL input directly to ground, if it must be held in a low state.



A	B	y
0	0	0
0	1	0
1	0	0
1	1	1

Figure 7-3

Unused inputs should always be tied to the supply via a pull-up resistor for a logic 1, or ground for a logic 0.

It is even more important to hardwire unused inputs of CMOS devices, since CMOS inputs do not necessarily float high like TTL inputs. Consequently, you must tie **all** unused CMOS inputs to V_{DD} or V_{SS}, depending on whether they are to be held high or low, respectively.

Another typical error created by the inexperienced designer is to leave the inputs of a JK flip-flop floating when the device is to be used in its toggling mode, as in a ripple counter. If the floating inputs change state as a result of a glitch, the flip-flop is placed in a different mode, resulting in an improper count sequence. Again, the preventive medicine is to tie the inputs to the +V supply via a pull-up resistor.

PROPAGATION DELAY

Propagation delay problems result from the accumulated delay of several gates in a signal path. The simple circuit and timing diagrams in Figure 7-4 illustrate this problem. Here, there are two signal paths to the two AND gate inputs. One path is connected directly to the AND gate, while the other path goes through a series of inverters before reaching the gate. Now, suppose the beginning of the signal paths are tied together as shown, and logic 1 pulse is applied. Ideally, without considering propagation delay, the AND gate output should go high. However, by the time the lower signal reaches the gate, the upper signal has returned to a logic 0 state. Thus, the output remains low as indicated by the associated timing diagram.

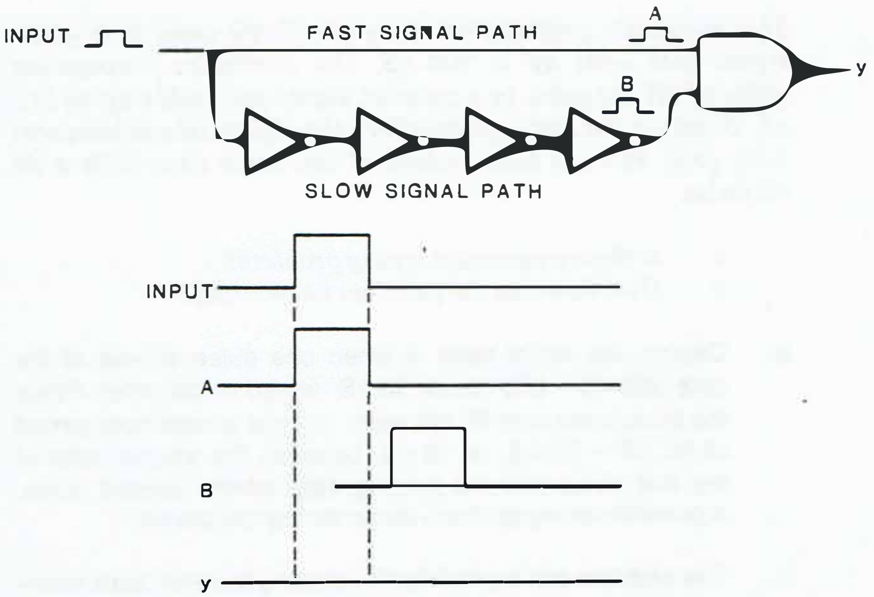


Figure 7-4
The propagation delay problem.

The timing problem in Figure 7-4 is created by the accumulated propagation delay of the inverters. Of course, your design would probably not employ four inverters in series, but you get the idea. Especially as circuits get more complex, the cummulative effect of propagation delay can create severe timing problems.

Once a design is finalized, propagation delay problems are very hard to isolate. This is because the individual gates operate properly when tested, but "for some reason" the overall circuit does not operate as intended. You should suspect propagation delay problems whenever you hear yourself say "everything is right, but **for some reason** the circuit just doesn't work".

There are several "tricks" you can use if you suspect propagation delay problems. In most cases, these problems are created when one signal path has many more gates than another as you observed in Figure 7-4. Thus, one solution is to slow-down the signal in the faster path by adding buffers or back-to-back inverters. However, this can be tricky business. First, you are not sure how much the faster signal should be delayed. To get a rough estimate, you can add-up all the maximum propagation delay times for the slower signal path and subtract all the minimum propagation delay times for the faster signal path. This will give you a **worst case** compensation value. You then add the proper number of buffers or inverters to the faster signal path to compensate for this amount of delay.

Example 7-1

The maximum propagation delay of all the gates in a given signal path adds up to 300 nS. The minimum propagation delay of all the gates in a parallel signal path adds up to 270 nS. Suppose the signal paths drive the inputs of a subsequent 2-bit gate, and are both pulsed at the same time with a 20 nS pulse.

a. Is there a potential timing problem?
b. If so, how can the problem be rectified?

a. Clearly, the worst case is when one pulse arrives at the gate 300 nS – 270 nS, or 30 nS, ahead of the other. Since the pulses are only 20 nS wide, there is a maximum period of 30 nS – 20 nS, or 10 nS, between the trailing edge of the first pulse and the leading edge of the second pulse. A possible timing problem exists during this period.

b. The problem can be rectified by placing back-to- back inverters in the faster signal path. You find that a 74LS04 has a typical propagation delay of 5 nS. As a result, 30 nS/5 nS, or six, back-to-back 74LS04's should compensate for the timing difference.

It is important to note that this compensation technique is based on minimum, maximum, and "typical" values of propagation delay. Unfortunately, you can't determine the precise amount of timing difference and required compensation, since propagation delay is so device dependent. In addition, external factors, such as temperature, affect this value. This is why I said earlier that delay compensation is tricky business. Sometimes, the best solution is simply a trial and error approach. That is, begin adding a delay using a worst case "ball-park" compensation value. If this doesn't work, add or subtract buffers or inverters until the timing problem is eliminated.

Before leaving this topic, I should also mention that propagation delay problems become even more prevalent in circuits that mix logic families, in particular TTL and CMOS logic. As you know, CMOS is much slower than TTL.

Sequential Logic Bugs

Sequential logic can suffer from the same set of design ills as combinational logic. Noise, power supply glitches, floating inputs, and propagation delay problems also apply to sequential circuits. For that reason, the same precautions should be taken to reduce these problems. However, one of the biggest problems in the design of sequential circuits is **timing**. Timing problems usually result from **unclocked inputs, clock skew**, and **excessive clock rates**.

UNCLOCKED INPUTS

The basic component of any sequential circuit is the flip-flop. Although the data inputs to a flip-flop are clocked, most flip-flops have unclocked, or **asynchronous**, PRESET and CLEAR inputs. These inputs can be used in basically two ways: for circuit initialization, or to set and clear the flip-flop during its sequential operation.

All flip-flop circuits must be designed with a means to initialize the circuit. This usually comes in the form of a Master Reset or Power On Reset circuit. The function of this circuit is to generate a pulse that will set and/or clear all the critical circuit flip-flops to an initial starting condition. The initialization pulse must be generated when the circuit is "powered-up" or "hangs-up", in a particular state. Hang-up conditions are another problem with sequential circuits and will be discussed shortly. Circuit initialization does not generally cause a timing problem, since the function of the operation is to abort a given sequence and begin again.

Timing problems appear when the PRESETS and CLEARS are activated asynchronously as part of the sequential operation of the circuit. To illustrate this, consider the BCD counter circuit in Figure 7-5. Here, the counter can be initialized to a 0000 state by activating the RESET line. In addition, during the count sequence, the counter is cleared to 0000 immediately after a count of 1001. On the tenth clock pulse, **Q** = 1010, making Q3 and Q1 both high. This causes a low output from the NAND gate which clears all the flip-flops and causes the count to roll-over to 0000.

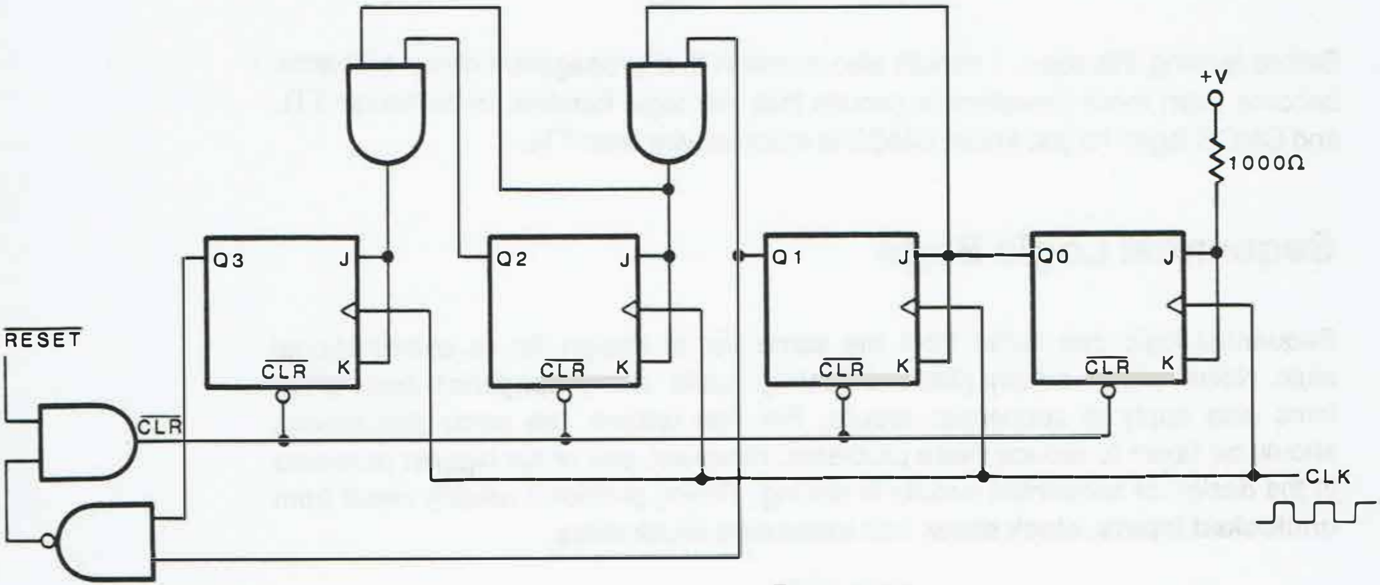


Figure 7-5

A synchronous BCD counter.

Now, consider the case when the count goes from 0111 to 1000 on the eighth clock pulse. All the flip-flops must toggle when this eighth clock hits. Suppose that the Q1 flip-flop has a longer propagation delay than the Q3 flip-flop. If this is the case, Q1 toggles after Q3. Thus, for an instant, both Q1 and Q3 are high, resulting in a short glitch on the CLR line as shown by the timing diagram in Figure 7-6. If this happens, the circuit could reset to 0000 at this point in the count, before even reaching its maximum count of 1001.

The solution to this glitch problem is to synchronize the $\overline{CLR}$ signal with the clock as shown in Figure 7-7. Here, an OR gate has been added so that the glitch in Figure 7-6 will not clear the circuit. Notice that the clock is high during the glitch period. This holds the OR gate output high, thereby preventing the glitch from clearing the circuit. By the time the clock goes low, the glitch is gone. However, when the counter must roll-over from 1001 to 0000, the NAND gate output remains low during the entire clock cycle. This low output is passed by the OR gate to the $\overline{CLR}$ inputs only when the clock goes low during the second half of its cycle.

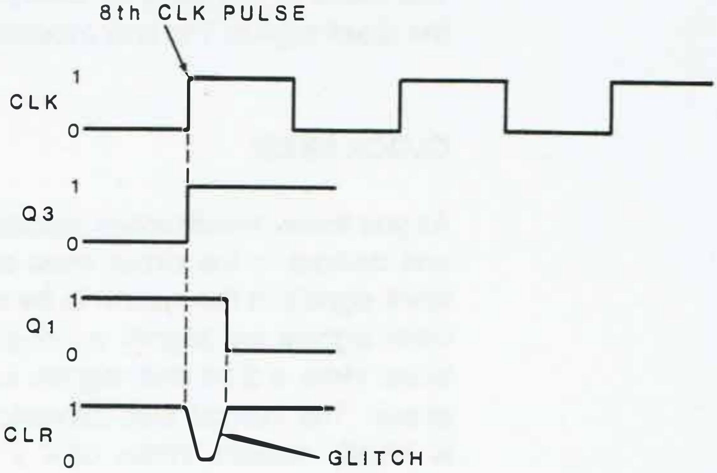


Figure 7-6

A short glitch might appear on the CLR line between a count of 0111 and 1000 for the BCD counter in Figure 7-5.

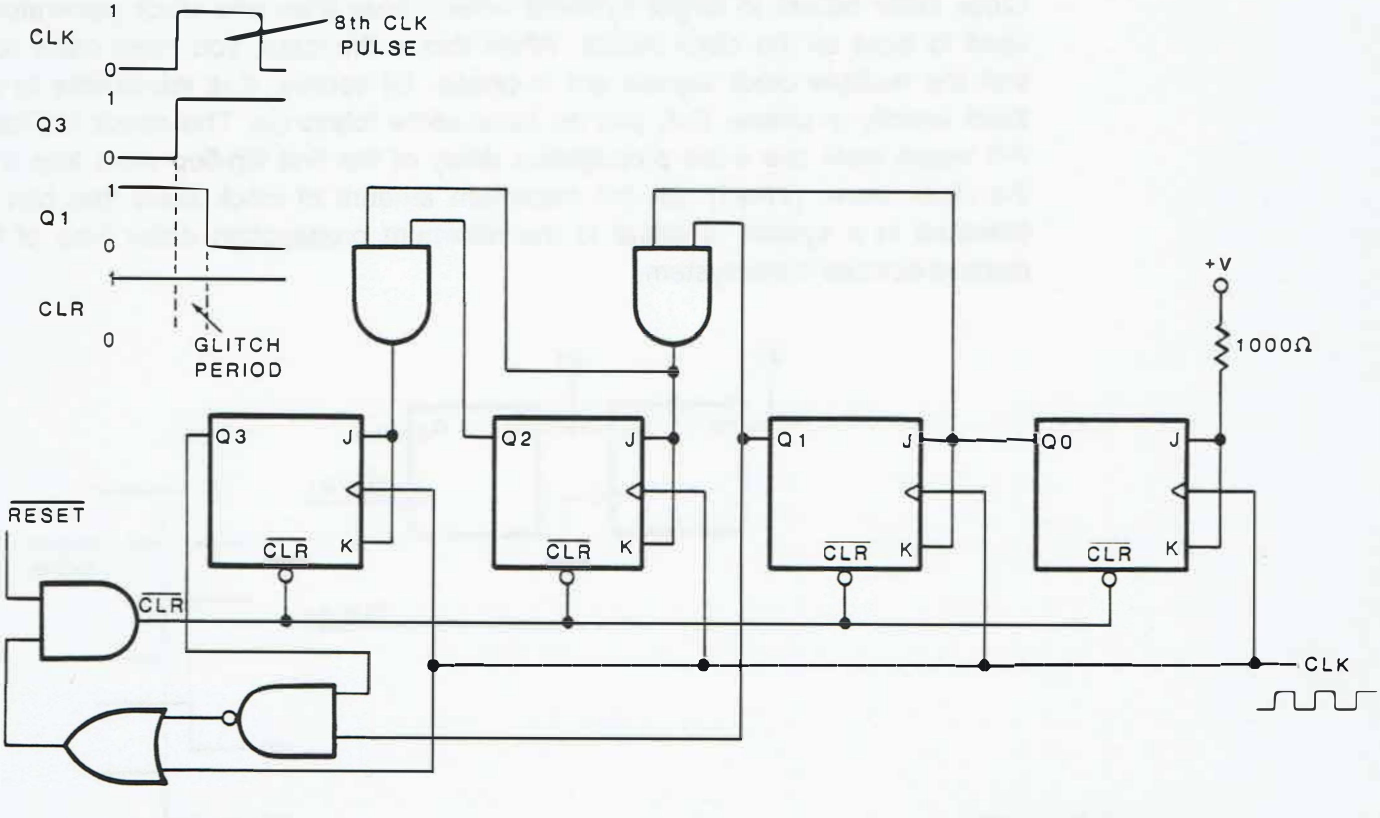


Figure 7-7

Synchronizing the CLR function with the clock signal.

The moral of this story is: **always synchronize PRESETS and CLEARS with the clock signal.** The only exception to the rule is for circuit initialization.

CLOCK SKEW

As you know, synchronous circuits operate on a clocking principle, whereby several devices in the circuit must be clocked at the same time. This requires all clock signals in the system to be in-phase. Clock skew occurs when two or more clock signals are slightly out-of-phase. Figure 7-8 illustrates the result of clock skew. Here, a 2-bit shift register is clocked by two signals that are slightly out-of-phase. The normal shift sequence should be 00, 01, 11. Suppose the circuit is initially cleared. When CLK #1 hits, Q0 goes high and Q1 stays low. You might think that everything is fine. However, this state is very short-lived, since when CLK #2 hits, Q1 will go high. This is not supposed to happen until the next clock cycle. However, because CLK #2 occurs slightly after CLK #1, the second flip-flop is already set-up with a logic 1 from Q0 by the time CLK #2 hits. The result is that the circuit will appear to shift from 00 to 11, skipping the 01 state.

Clock skew occurs in larger systems where more than one clock generator is used to drive all the clock inputs. When this is the case, you must make sure that the multiple clock signals are in-phase. Of course, it is impossible to get them exactly in-phase. But, you do have some tolerance. The circuit in Figure 7-8 would work fine if the propagation delay of the first flip-flop were less than the clock skew. (Why?) So, the maximum amount of clock skew that can be tolerated in a system is equal to the minimum propagation delay time of the clocked devices in the system.

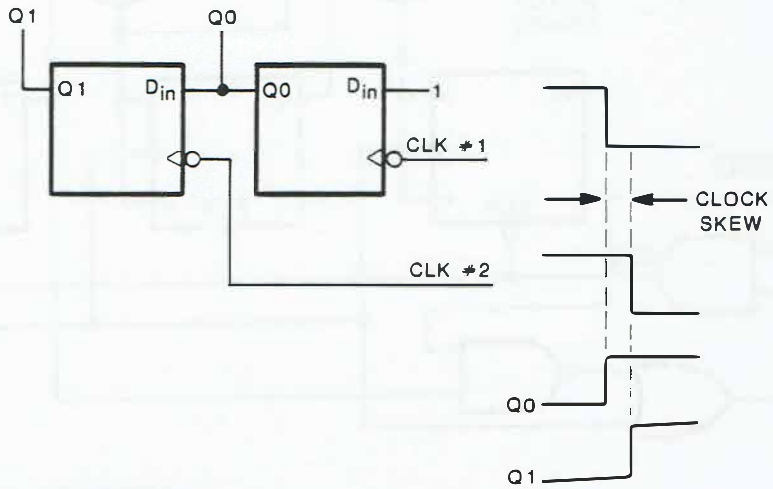


Figure 7-8

A 2-bit shift register clocked by two out-of-phase clock signals.

Even if you do manage to adjust two clock generators to be in-phase within the above limits, the signals will tend to drift with temperature and age, thereby requiring frequent adjustment. The best solution is to use a single clock generator. However, even with a single clock generator, the clock skew problem can still be present due to excessive **wiring delay**. It takes about 1.5 nanoseconds for a clock signal to travel one foot. If two devices where clocked at opposite ends of a long clock signal line, this wiring delay might be enough to create a clock skew problem. This is especially true of very fast logic families, such as ECL. The solution to this problem is to keep the clock signal lines as short as possible, making sure there are no unnecessary loops in the signal path.

EXCESSIVE CLOCK RATES

The clocking rate of a sequential circuit cannot exceed the amount of time that it takes the circuit to respond to the clock pulse and set-up for the next clock pulse. Basically, this amount of time is equal to the sum of the propagation delays and set-up times in a signal path. For example, consider the circuit in Figure 7-9. Once the circuit is clocked, another clock pulse cannot arrive until the data signal has propagated through the first flip-flop, through the two gates, and satisfied the set-up time of the second flip-flop. Thus, the minimum clock period in Figure 7-9 must be:

Min. Clock Period = 1st Flip-Flop Delay + 1st Gate Delay +
2nd Gate Delay + 2nd Flip-Flop Set-Up Time

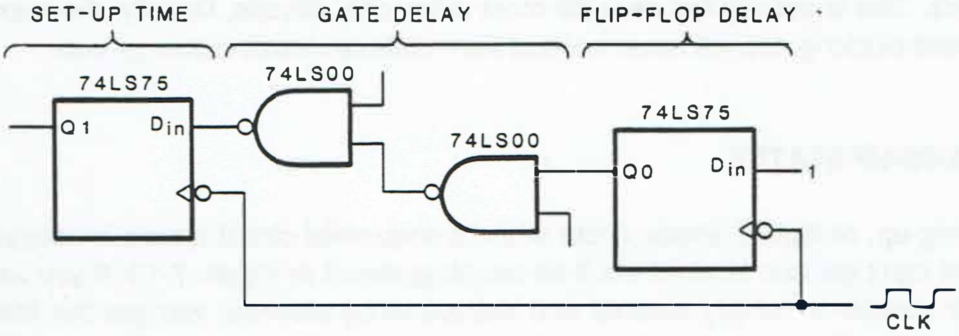


Figure 7-9

The maximum clock frequency is a function of flip-flop delay, gate delay, and set-up time.

Example 7-2

A 74LS00 specifies a maximum propagation delay of 10 nS. A 74LS75 specifies a maximum propagation delay of 27 nS and a minimum set-up time of 20 nS. What is the maximum frequency at which the circuit in Figure 7-9 can be clocked using these devices?

The circuit in Figure 7-9 has one flip-flop propagation delay, two gate delays, and one set-up time. Using a 74LS00 and 74LS75, the minimum clock period is:

$$\begin{aligned}\text{Min. Clock Period} &= 1 \text{ flip-flop delay} + 2 \text{ gate delays} + 1 \text{ set-up time}\\ &= 27\text{ nS} + (2 \times 10\text{ nS}) + 20\text{ nS}\\ &= 67\text{ nS}\end{aligned}$$

Thus, the maximum clock frequency is:

$$\begin{aligned}\text{Max. Clock Freq.} &= 1/\text{Min. Clock Period}\\ &= 1/67\text{ nS}\\ &= 14.9\text{ MHz}\end{aligned}$$

It is doubtful that you would ever need to clock the circuit in Figure 7-9 (Example 7-2) at such a high rate. However, you should be aware that a limit exists, and that this limit decreases as the number of flip-flop stages increases and as the more logic is added between flip-flop stages.

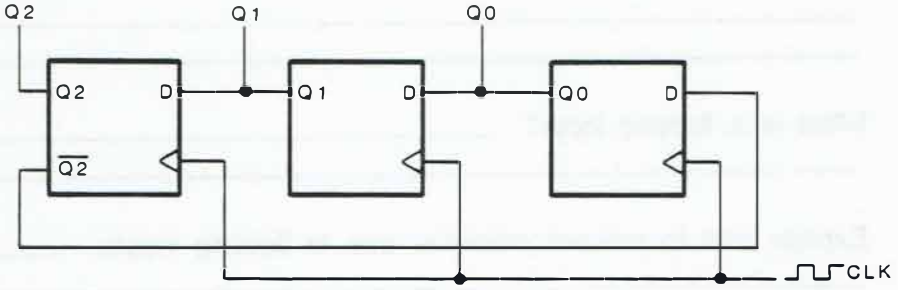
I should also mention that there is a maximum frequency at which individual devices can be clocked. For instance, the 74LS75 used in the above example has a maximum clock frequency around 30 MHz. Of course, that doesn't pose any problems in the circuit in our example, since 14.9 MHz is well below 30 MHz. This is usually the case for most sequential circuits. Usually, the maximum circuit clocking rate will never exceed the individual device clocking rates.

HANG-UP STATES

Hang-up, or locked, states occur when a sequential circuit enters an illegal state and can't get out. Look at the 3-bit counting circuit in Figure 7-10. If you assume the counter is initially cleared and subsequently clocked, you get the following count sequence:

0 0 0
0 0 1
0 1 1
1 1 1
1 1 0
1 0 0
0 0 0
.
.
.

This sequence shows that there are six legal states. Two states, 010 and 101, are not included and therefore are undefined for this counter. Consequently, they should never occur, right? But, what happens if, for some unknown reason, they do occur? Suppose some external condition, such as noise, puts the counter into the 010 state. What state will the counter generate next? —— Tracing the flip-flop logic in Figure 7-10, the next state after 010 would be the other illegal state, 101. What is the next state after this? —— Again, following the circuit logic the next state after 101 is the first illegal state, 010. As you can now see, the circuit sequence would be 010, 101, 010, 101, 010, etc. In other words, the circuit is locked-up in these two illegal states.



STATE	Q2Q1Q0
0	0 0 0
1	0 0 1
2	0 1 1
3	1 1 1
4	1 1 0
5	1 0 0
6	0 0 0
ILLEGAL	0 1 0
	1 0 1

Figure 7-10

A poorly designed sequential circuit can hang-up in illegal states.

The only way to release a circuit from a hang-up state is to re-initialize it. This is why it is important that all sequential circuits include a reset function. In addition, the circuit must be designed so that it will automatically go back to the legal count sequence should any illegal state occur. To do this, you must make sure that **all** possible states are accounted for in the state table during the design of the circuit. You might recall from Unit 4 that illegal states where included in the state table and used as don't cares during the sequential circuit design process. This practice will always assure you that the circuit will automatically go back to the legal sequence should any illegal state occur.

Self-Test Review

1. List the four most common categories of combinational logic design bugs.

A. ______________________________
B. ______________________________
C. ______________________________
D. ______________________________

2. Explain how to prevent power supply glitches from affecting a digital IC.

____
____

3. What is a floating input? ______________________________

____

4. Explain how to prevent problems due to floating inputs. __________

____
____

5. How do propagation delay problems occur in combinational logic circuits?

____
____
____

6. Name the three potential timing problems in sequential circuits.

A. ______________________________
B. ______________________________
C. ______________________________

7. Explain how to prevent timing problems due to unclocked inputs in a sequential circuit. ______________________________

8. How much clock skew can be tolerated in a digital system? ______________________________

9. A 74LS76 JK flip-flop has the following specifications:

	Minimum	**Typical**	**Maximum**
t_{pLH}	—	11 nS	20 nS
t_{pHL}	—	16 nS	20 nS
$t_{SET\text{-}UP}$	20 nS	10 nS	—
f_{MAX}	30 MHz	45 MHz	—

A 74LS02 NOR gate has the following specifications:

	Minimum	**Typical**	**Maximum**
t_{pLH}	—	5 nS	10 nS
t_{pHL}	—	5 nS	10 nS

What is the maximum clock frequency possible if two 74LS76 flip-flops are separated by three 74LS02 gates in a sequential circuit?

10. Explain how to prevent hang-up states in sequential circuits. ______________________________

Answers

1A. Noise

1B. Power Supply Glitches

1C. Floating Inputs

1D. Propagation Delay

2. By placing a 0.01 μF to 0.05 μF capacitor across the +V and ground pins of the IC.

3. It is an unconnected input that floats-up to a logic 1 state.

4. Tie all unconnected inputs to the +V supply via a 1000 ohm pull-up resistor if they are to remain at a logic 1 state. Tie unconnected inputs directly to ground if they are to remain in a logic 0 state.

5. They occur when one signal path is much faster than another path and the two paths are used to drive a common set of inputs.

6A. Unclocked inputs

6B. Clock skew

6C. Excessive clock rates

7. They can be prevented by synchronizing all PRESETs and CLEARs with the clock signal.

8. The maximum amount that can be tolerated is equal to the minimum propagation delay time of the clocked devices in the system.

9. 14.286 MHz

10. Make sure that all possible states are accounted for in the state table during the design process.

Following is the solution to question 9.

9. Min. CLK Period = 1st flip-flop delay + all gate delays + 2nd flip-flop set-up time
= 20 nS + (3 × 10 nS) + 20 nS
= 70 nS

Max. CLK Freq. = 1/Min. CLK Period
= 1/70 nS
= 14.286 MHz

TEST INSTRUMENTS

Once you have implemented your design in prototype form, you are sure to have some circuit bugs that can't be detected by theoretical means. This is when you must get your hands dirty and troubleshoot the design to isolate the bugs. In this section, you will get acquainted with several test instruments that are commonly used to troubleshoot digital circuits. Then, in the next section, you will learn how to troubleshoot circuits using these instruments. I will begin with the simplest and most common digital test instruments, then progress to more sophisticated instruments.

I should caution you, however, that there is no substitute for experience when working with test instruments and troubleshooting circuits. The following material is provided as a guide to make you aware of what instruments are available and how they are used for digital circuit troubleshooting. The rest is up to you!

Logic Probes

The logic probe is the simplest and probably the most useful test instrument for digital circuit troubleshooting. A typical logic probe is pictured in Figure 7-11. The basic function of the probe is to allow you to trace 1's and 0's through the circuit under test. Notice that there are LED indicators for HIGH and LOW that illuminate when a logic 1 or logic 0 are detected, respectively.

To use the probe, you must first set the logic family switch to either TTL or CMOS, depending on what type of logic is used in your circuit. Next, you must connect the two leads to the power supply of the circuit being tested. In most cases the leads are color coded red and black. The red lead is connected to the +V supply and black lead to ground. You are then ready to probe the circuit while observing the LED indicators to determine the logic at various points in the circuit.

Most logic probes, like the one in Figure 7-11, have an LED indicator labeled PULSE and an associated MEMORY/PULSE switch. When the switch is in the PULSE position, the PULSE LED will illuminate for a few tenths of a second each time the input logic changes state. Thus, the PULSE LED would blink on and off if a clock signal were probed. In addition, the HIGH and LOW LED indicators would alternately blink or seem constantly illuminated, depending on the clock frequency.

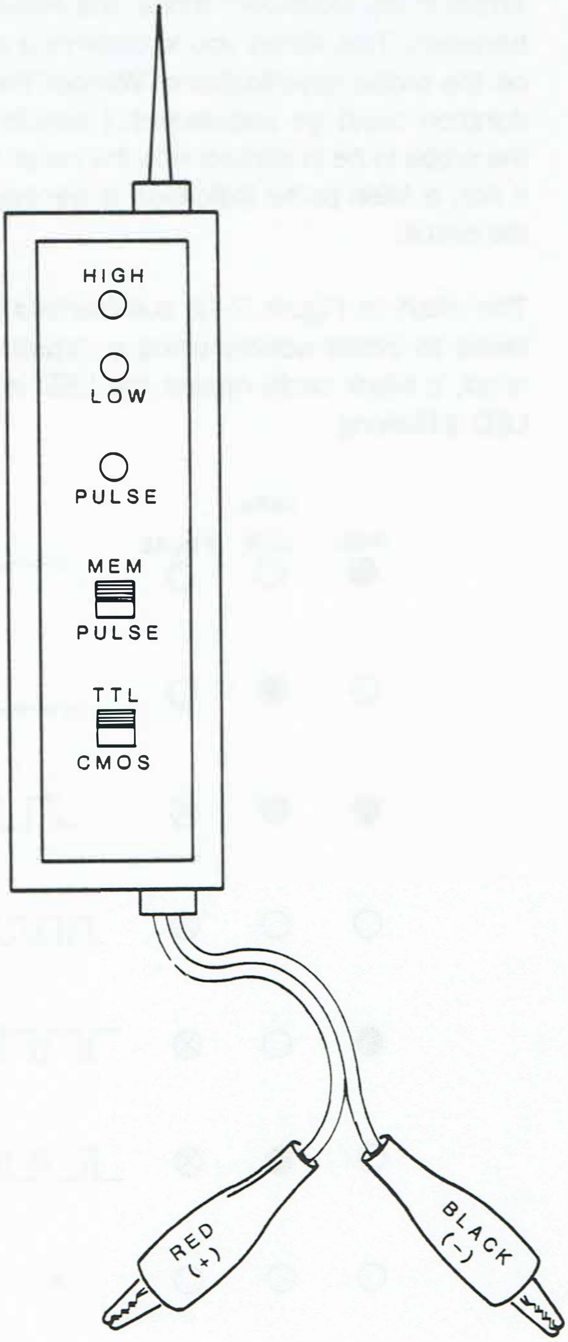


Figure 7-11
A typical logic probe.

When in the MEMORY mode, the PULSE LED will stay illuminated after a logic transition. This allows you to observe a single pulse as short as 10 nS, depending on the probe specifications. Without the memory feature, a pulse of such short duration could go undetected. I should mention that most logic probes require the probe to be in contact with the circuit before the MEMORY feature is activated. If not, a false pulse indication is generated when the probe makes contact with the circuit.

The chart in Figure 7-12 summarizes different LED indications and how they relate to circuit activity using a "typical" probe. A blank circle means the LED is off, a black circle means the LED is on, and a circle with an "X" means the LED is blinking.

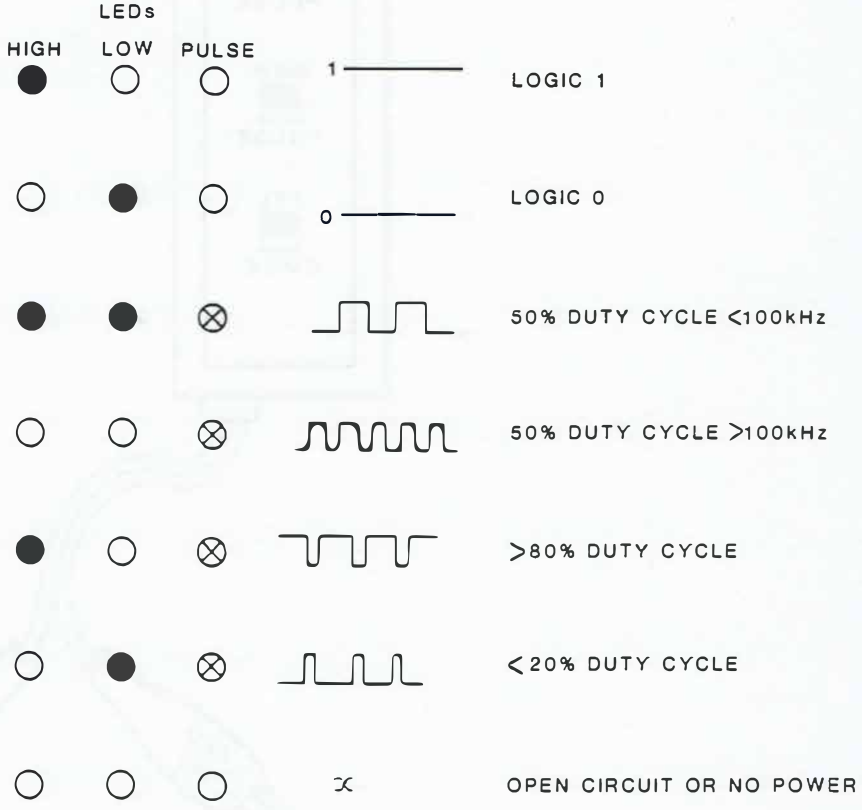


Figure 7-12

A summary of logic probe indications.

Logic Clips

A logic clip, like the one in Figure 7-13, is a simple mechanical device that allows easier access to the pins of an IC. The logic clip has two rows of contacts and is spring loaded so that it can be clamped on the pins of a dual in-line, or DIP, package as shown. The IC pins are then tested by probing the contacts that are brought-out on the top of the clip. Logic clips are commonly available to fit 14-, 16-, and 24-pin ICs.

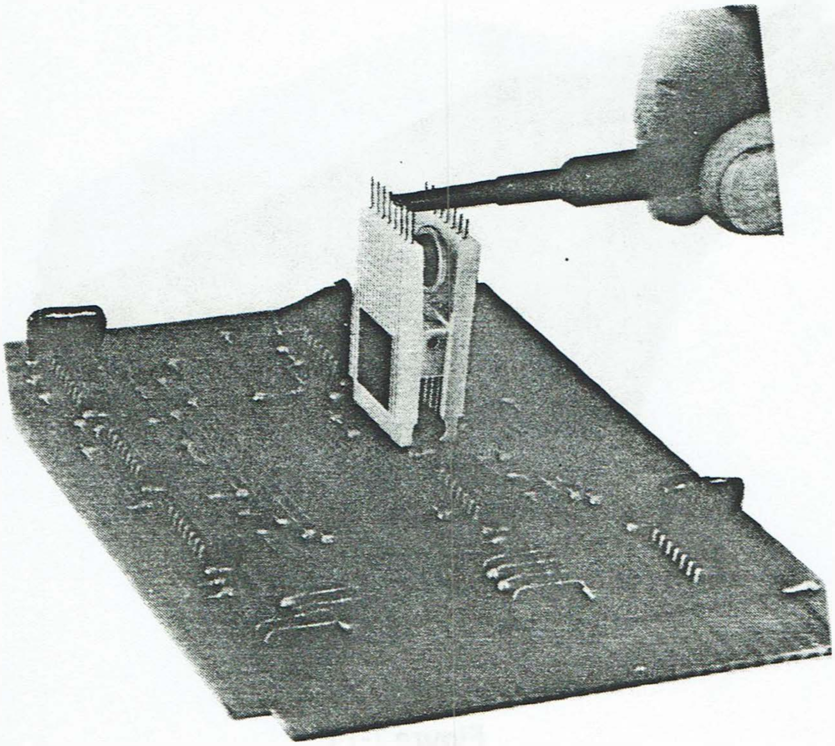


Figure 7-13

A logic clip extends the IC signals.

(Photograph courtesy AP Products.)

Logic Monitors

The logic monitor, shown in Figure 7-14, is a fancy test clip with built-in logic indicators. The monitor is clamped on an IC and the LED indicators provide a visual display of the logic on each of the IC pins simultaneously. The advantage of using a logic monitor over a logic probe is speed. With a logic monitor, you do not have to probe the IC pins individually. The monitor gives you a very quick check of an IC to determine if a problem exists or not. However, a logic monitor does not have any pulse or memory features. Consequently, it can only be used to observe stationary or slowly changing logic signals.

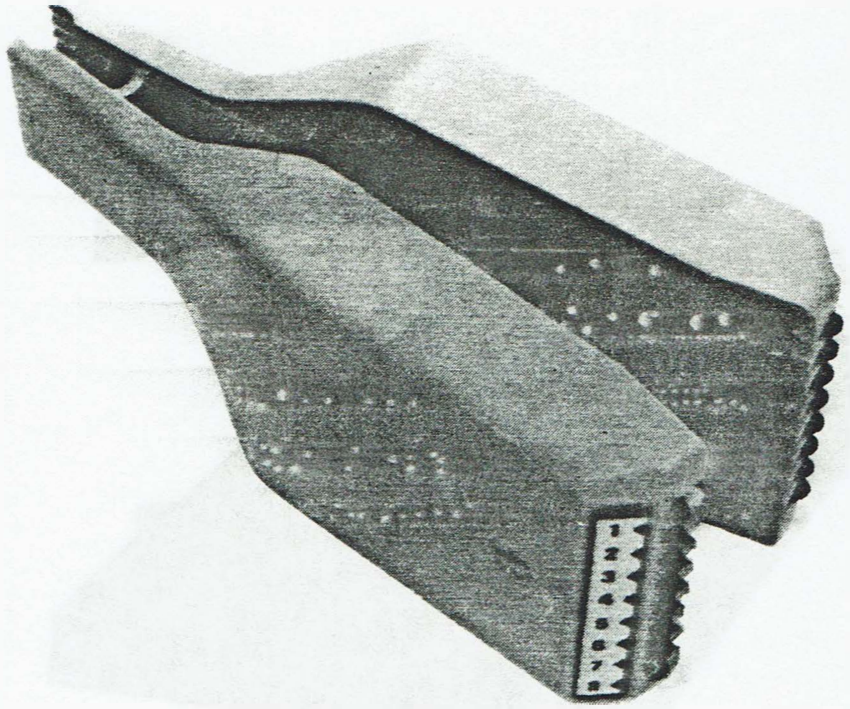


Figure 7-14

A logic monitor provides LED indications of all the IC logic simultaneously.

(Photograph courtesy Global Specialties.)

Logic Pulsers

Although the logic pulser in Figure 7-15 looks like a logic probe, it performs just the opposite function. A logic pulser transmits, rather than receives, logic pulses. The logic pulser is connected to the circuit power supply just like the logic probe. You must then select between TTL or CMOS, depending on the type of logic in the circuit being tested. You then place the probe tip in the circuit and push the PULSE button to generate a pulse. The logic pulser automatically senses which logic level is present and generates the opposite logic state. For instance, if the circuit point is a logic 0, the pulser generates, or sources, a logic 1. Conversely, if the circuit point is a logic 1, the pulser sinks the circuit to a logic 0.

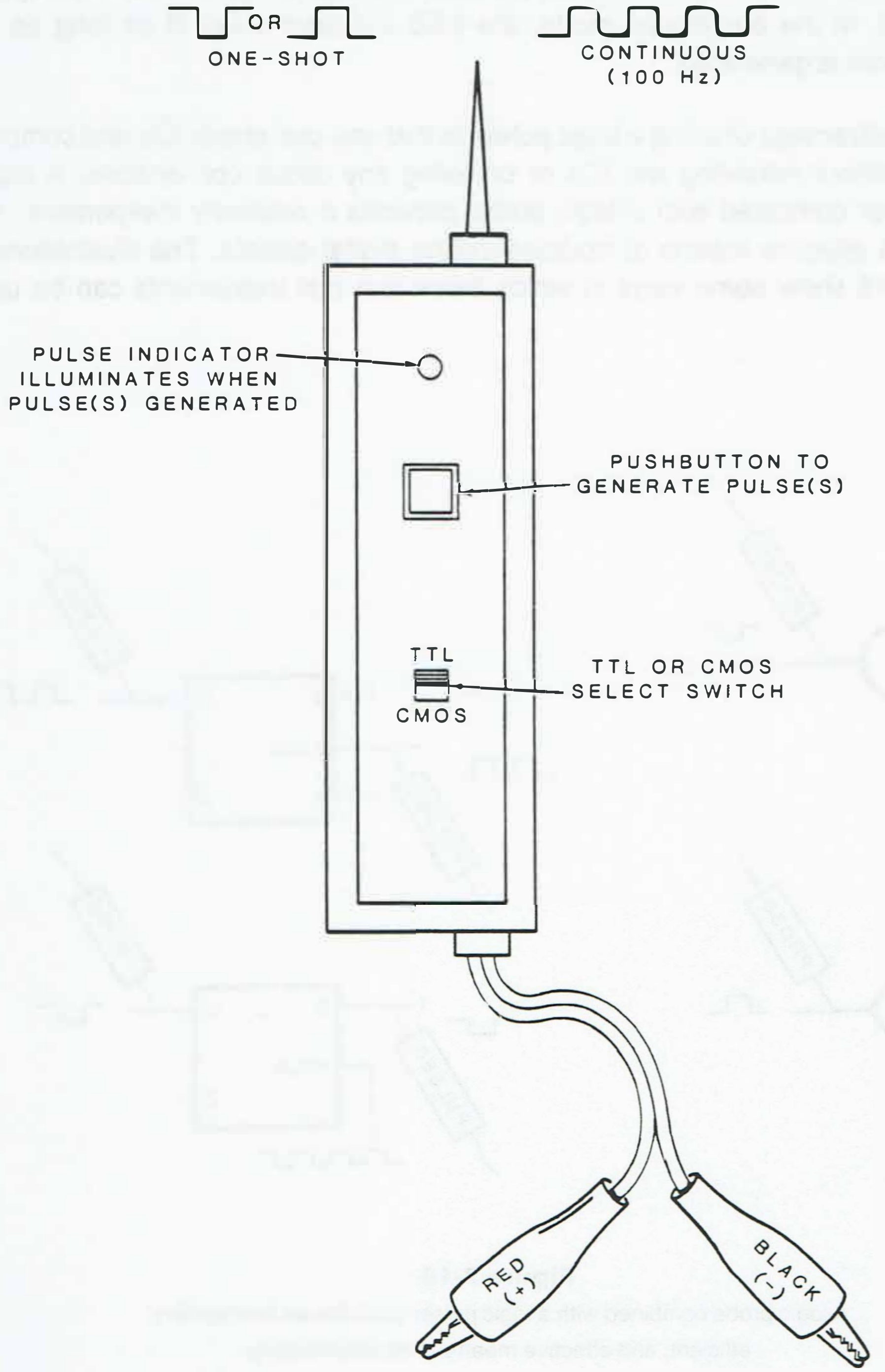

OR
ONE-SHOT
CONTINUOUS (100 Hz)
PULSE INDICATOR ILLUMINATES WHEN PULSE(S) GENERATED
PUSHBUTTON TO GENERATE PULSE(S)
TTL
CMOS
TTL OR CMOS SELECT SWITCH
RED (+)
BLACK (−)

Figure 7-15
A typical logic pulser.

Most logic pulsers can generate a one-shot (single) pulse, or a continuous stream of pulses. One popular model generates a single one-shot pulse when the PULSE button is pushed less than one second. If the PULSE button is held over one second, a continuous 100 Hz stream of pulses is generated as long as the button is depressed. In the one-shot mode, the LED indicator blinks once for each pulse generated. In the continuous mode, the LED indicator stays lit as long as the pulse stream is generated.

The real advantage of using a logic pulser is that you can check ICs and complete circuits without removing any ICs or breaking any circuit connections. A digital logic pulser combined with a logic probe provides a relatively inexpensive, efficient, and effective means of troubleshooting digital circuits. The illustrations in Figure 7-16 show some ways in which these two test instruments can be used together.

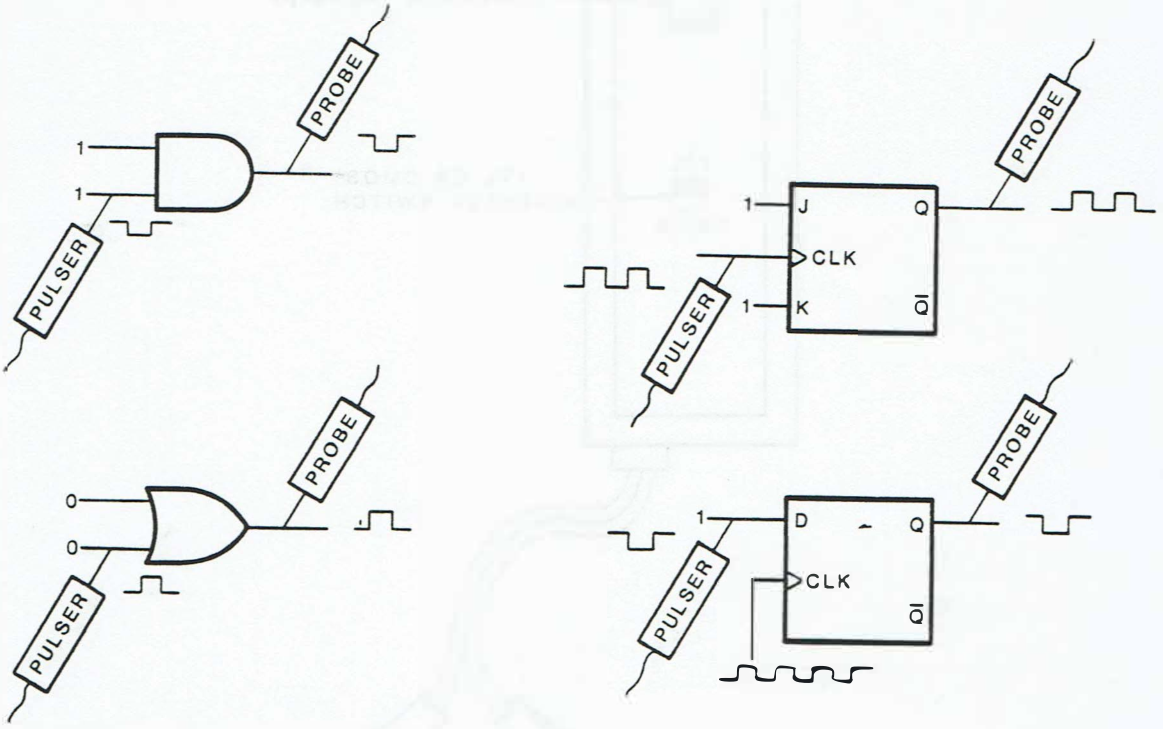


Figure 7-16

A logic probe combined with a logic pulser provides an inexpensive, efficient, and effective means of troubleshooting.

Multimeters

The name "multimeter" comes from the fact that these test instruments contain several different measuring instruments in one package. Most multimeters are capable of measuring AC and DC voltage, AC and DC current, and resistance. Several different multimeters are pictured in Figure 7-17. Observe that are two general types: analog and digital. Analog multimeters have an analog meter movement (Figure 7-17a), while digital multimeters, or DMMs, have a digital display (Figures 7-17b).

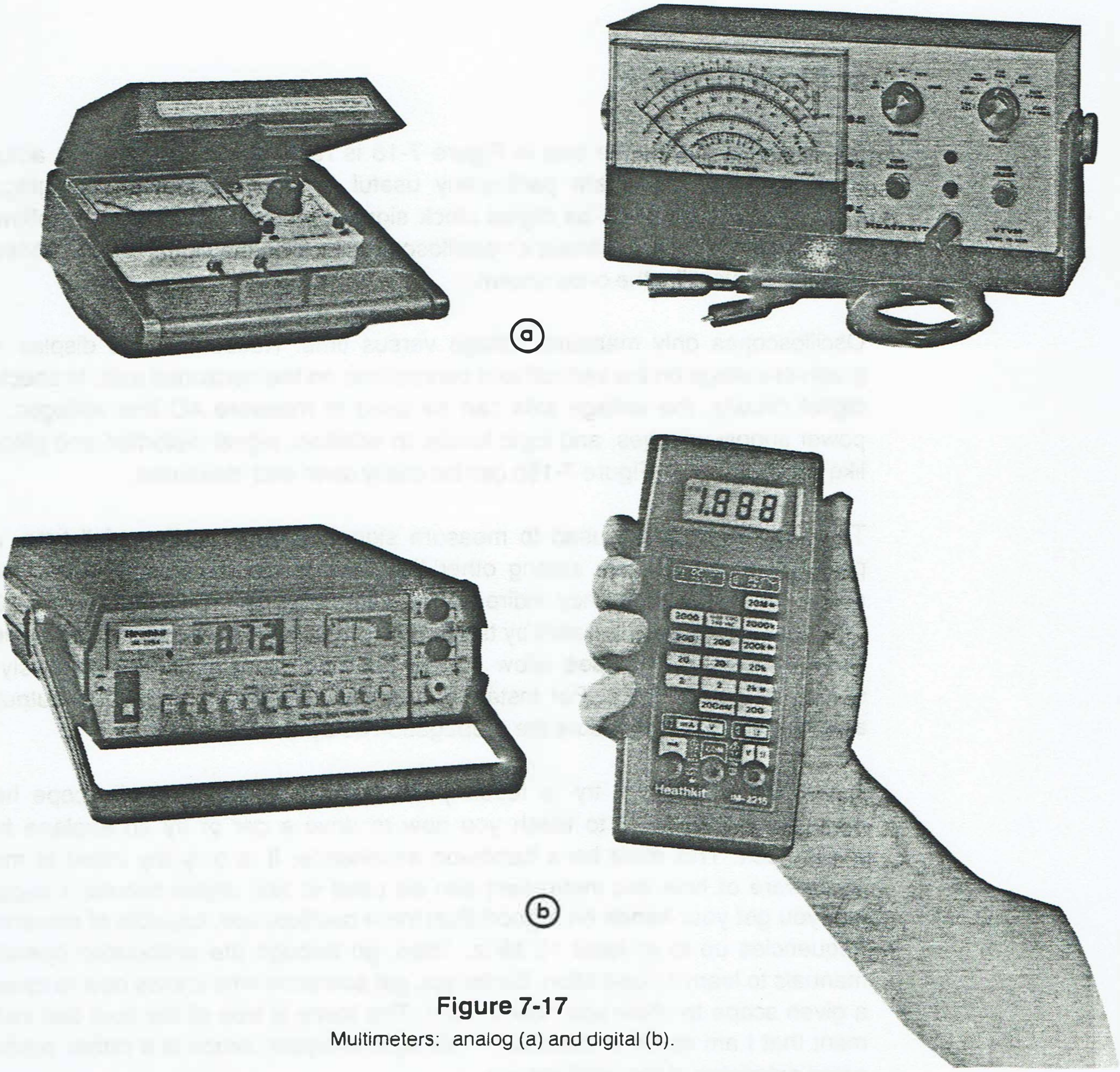


Figure 7-17

Multimeters: analog (a) and digital (b).

Multimeters are useful for testing AC line voltages and DC power supply voltages in digital circuits. In addition, static, or stationary, logic levels can be accurately measured to see if they are within their prescribed limits. When checking these DC voltage levels, a digital multimeter is preferred over an analog meter, since the DMM generally provides more accurate readings.

Possibly the most useful feature of a multimeter is its resistance measurement ability. Resistance measurements allow you to to check for shorts and opens in conductor paths. This is called **continuity checking**. In addition, you can check the quality of discrete components such as resistors, diodes, and transistors using the resistance range of a multimeter.

Oscilloscopes

An oscilloscope like the one in Figure 7-18 is required when you must actually "see" a signal. They are particularly useful for measuring fast changing, or **dynamic**, signals such as digital clock signals. A multimeter cannot "follow" a fast changing signal, whereas an oscilloscope can lock-on the signal and generate a visual display like the ones shown.

Oscilloscopes only measure voltage versus time. Notice that the display is a graph of voltage on the vertical axis versus time on the horizontal axis. In checking digital circuits, the voltage axis can be used to measure AC line voltages, DC power supply voltages, and logic levels. In addition, signal distortion and glitches like those shown in Figure 7-18b can be easily seen and measured.

The time axis can be used to measure signal period, rise time, fall time, and propagation delay time, among other time-related measurements. The oscilloscope measures frequency indirectly by measuring period (See Figure 7-18a). Frequency must be calculated by taking the reciprocal of the period measurement. **Dual trace oscilloscopes** allow you to display two signals simultaneously as shown in Figure 7-18c. For instance, you can display the input and output of a digital device and measure the propagation delay time as shown.

It makes no sense to try to teach you how to operate an oscilloscope here. It would be like trying to teach you how to drive a car or fly an airplane from a text book. This must be a hands-on experience. It is only my intent to make you aware of how this instrument can be used to test digital circuits. I suggest that you get your hands on a good dual trace oscilloscope, capable of measuring frequencies up to at least 10 MHz. Then, go through the associated operating manuals to learn its operation. Better yet, get someone who knows how to operate a given scope to show you how to do it. The same is true of the next test instrument that I am about to discuss — the logic analyzer, which is a rather sophisticated extension of the oscilloscope.

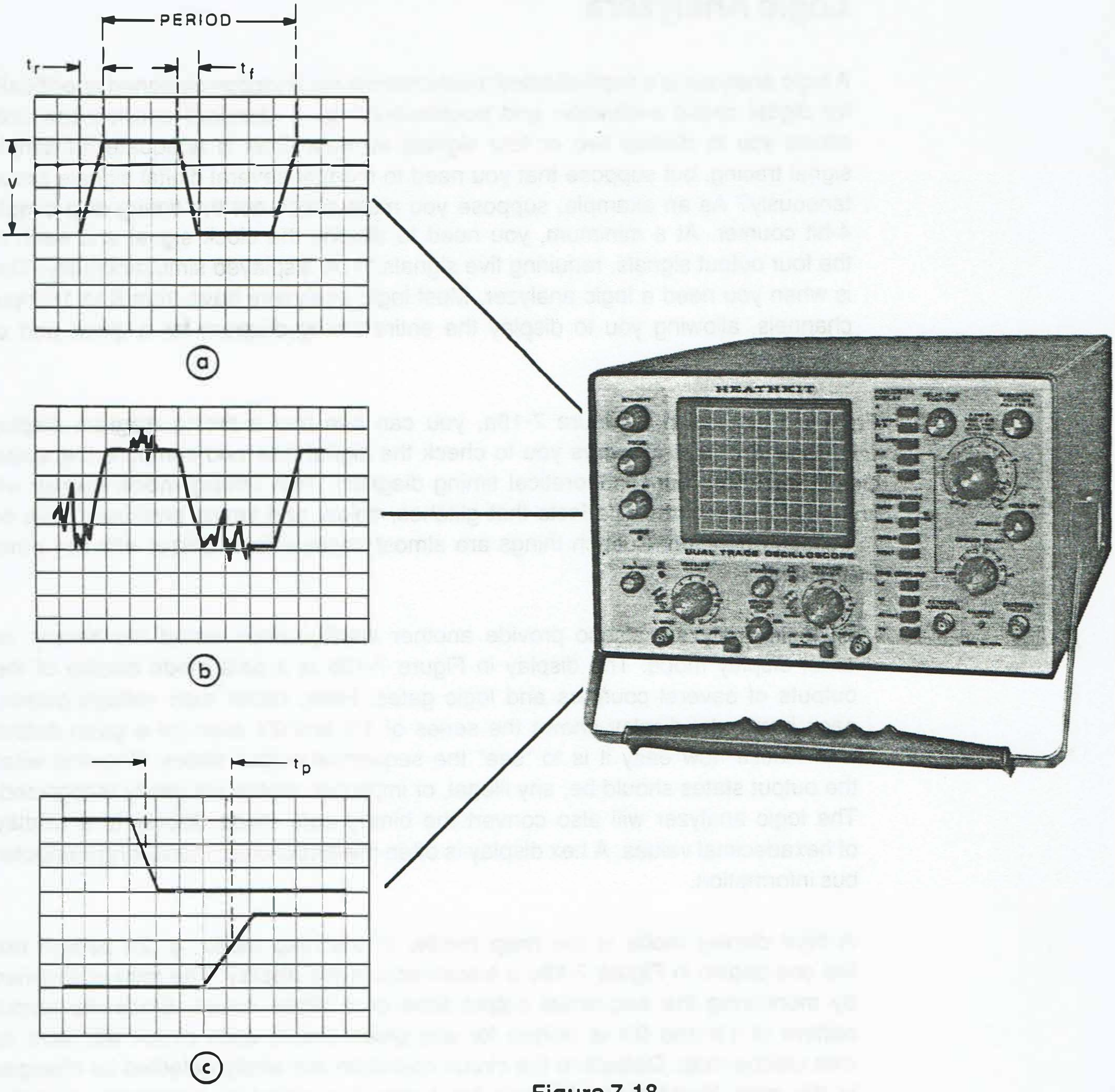


Figure 7-18

An oscilloscope allows you to "see" dynamic signals.

Logic Analyzers

A logic analyzer is a sophisticated multi-channel oscilloscope designed specifically for digital circuit evaluation and troubleshooting. A standard oscilloscope only allows you to display two or four signals at most. This is adequate for simple signal tracing, but suppose that you need to monitor several digital signals simultaneously? As an example, suppose you must check-out the timing of a simple 4-bit counter. At a minimum, you need to display the clock signal and each of the four output signals, requiring five signals to be displayed simultaneously. This is when you need a logic analyzer. Most logic analyzers have from 8 to 16 input channels, allowing you to display the entire timing diagram for a given part of the circuit.

From the display in Figure 7-19a, you can see that a timing diagram display on a logic analyzer allows you to check the signal flow and compare the actual circuit timing to your theoretical timing diagram. This **timing mode** display will immediately show the effects that glitches, noise, and timing problems have on the circuit operation. Such things are almost impossible to detect with any other instrument.

Most logic analyzers also provide another display mode called the **binary**, or **data**, display mode. The display in Figure 7-19b is a data mode display of the outputs of several counters and logic gates. Here, rather than voltage pulses, each horizontal display shows the series of 1's and 0's seen on a given output line. Notice how easy it is to "see" the sequential output states. Knowing what the output states should be, any illegal, or improper, states are easily recognized. The logic analyzer will also convert the binary data mode display to a display of hexadecimal values. A hex display is often preferred when monitoring computer bus information.

A third display mode is the **map mode**. In the map mode, a dot pattern like the one shown in Figure 7-19c is traced across the display. The map is obtained by monitoring the sequential output lines of a digital circuit. Since the output pattern of 1's and 0's is unique for any given circuit, each circuit will have its own unique map. Defects in the circuit operation are easily detected as changes in the map. Sometimes, the map for a circuit is called its **signature**. Just like you have your own unique signature, so does a sequential circuit. If a defect occurs, its signature will change.

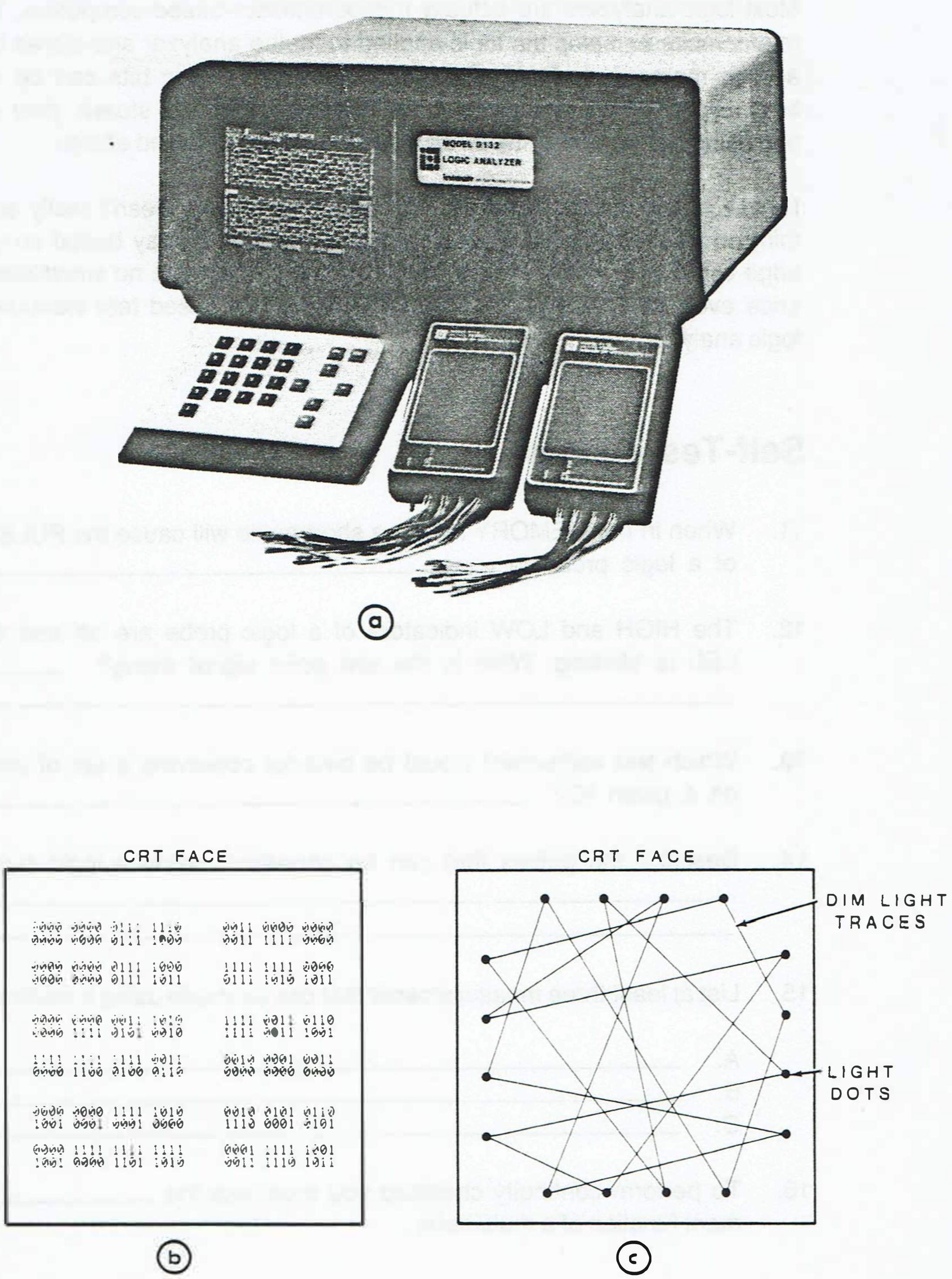


Figure 7-19

Model D132 Logic Analyzer showing timing mode display (a), data mode display (b), and map mode display (c).

(Photograph courtesy Intech.)

Most logic analyzers are actually microprocessor-based computers. The internal microprocessor samples the logic applied to the analyzer and stores it in random access memory, or RAM. Several thousands of logic bits can be stored at a time. Once the digital signals have been sampled and stored, they can be displayed using any one of the three display modes discussed above.

I should caution you, however, that a logic analyzer doesn't really analyze anything. **You** must still interpret, or analyze, a given display based on your knowledge of both the analyzer and the circuit. Again, there is no substitute for experience even when using a sophisticated computer-based test instrument like the logic analyzer.

Self-Test Review

11. When in the MEMORY mode, a short pulse will cause the PULSE indicator of a logic probe to ______________________________ .

12. The HIGH and LOW indicators of a logic probe are off and the PULSE LED is blinking. What is the test point signal doing? __________
____

13. Which test instrument would be best for observing a set of static signals on a given IC? ______________________________

14. Describe the pulses that can be generated using a logic pulser. __
____
____

15. List at least three measurements that can be made using a multimeter.

A. ____
B. ____
C. ____

16. To perform continuity checking you must use the __________ measurement function of a multimeter.

17. List at least three measurements that can be made on digital signals using an oscilloscope.

    A. ______________________________
    B. ______________________________
    C. ______________________________
    D. ______________________________
    E. ______________________________

18. Explain how to determine the frequency of a clock signal using an oscilloscope. ______________________________

    ____

19. Name the three display modes possible with a logic analyzer.

    A. ______________________________
    B. ______________________________
    C. ______________________________

20. Explain the basic operation of a microprocessor-based logic analyzer.

    ____
    ____
    ____

Answers

11. illuminate and remain illuminated.

12. The test point signal is switching at a rate greater than 100 kHz.

13. A logic monitor.

14. A single, or one-shot, pulse or a continuous 100 Hz stream of pulses.

15A. Resistance

15B. DC or AC voltage

15C. DC or AC current

16. resistance

17A. Logic levels

17B. Signal period

17C. Rise time

17D. Fall time

17E. Propagation delay time

18. Use the oscilloscope to measure the signal period then calculate its reciprocal (1/period).

19A. Timing mode

19B. Binary, or data, mode

19C. Map mode

20. The microprocessor samples the logic applied to the analyzer and stores it in RAM. Then the logic is displayed using any one of the three display modes.

CIRCUIT TROUBLESHOOTING AND REPAIR

Now that you are familiar with the basic tools used to troubleshoot digital circuits, it is time to use these tools to find circuit problems and get them corrected. It is impossible to provide you with a foolproof step-by-step procedure for circuit troubleshooting. However, I can give you some general guidelines and tips that have proven to be effective throughout the servicing industry. This section is devoted to a discussion of the following troubleshooting guidelines:

1. Obtain and study any available circuit documentation.
2. Use your senses and common-sense to look for the obvious.
3. Test the power supply circuit(s).
4. Test the clock circuit(s).
5. Troubleshoot the combinational logic.
6. Troubleshoot the sequential logic.
7. Make the repair.

Documentation

Troubleshooting a complex digital circuit without the associated documentation is like trying to make a trip to an unknown place without a road map — you often get lost. The documentation for a circuit can take many forms. If the circuit is a new design prototype, the documentation should include a black-box diagram and detailed schematic diagram at a minimum. Other available materials might be truth tables, state tables, timing diagrams, IC layout diagrams, wiring lists, and PC board conductor pattern diagrams. You will also need data sheets for the circuit ICs and discrete components if detailed troubleshooting is required.

If the circuit has been manufactured for some time on a commercial basis, you should be able to obtain operating and service manuals from the manufacturer. These manuals are extremely valuable, since they often include disassembly/reassembly guides, troubleshooting flowcharts and guides, service bulletins, parts lists, etc.

Sometimes, there is a service and maintenance record kept on previous repairs and periodic maintenance. This is especially true in the industrial environment. Many times, the same problems reoccur periodically due to poor design or improper maintenance. If this is the case, service records might provide an important clue as to the nature of the problem.

Even other individuals who have serviced the circuit before can be an important source of information. At a minimum, they can relate their experiences and give you some tips that might suggest the cause of the problem.

The idea behind documentation is to use whatever resources are available to you, from schematics to people, in order to locate the problem quickly and get it corrected. The check-list in Table 7-1 summarizes the various forms of documentation that might be available for a circuit. You might want to cut this out and keep it for future reference.

Table 7-1: Documentation Check List

- Schematic Diagrams
- Block Diagrams
- Truth/State Tables
- Timing Diagrams
- IC Layout Diagrams
- Wiring Lists
- PC Board Conductor Patterns
- IC Data Sheets
- Component Data Sheets
- Operating Manuals
- Service Manuals
- Service Records
- Maintenance Records
- Other Individuals

Once you have obtained all the necessary documentation, you must study it until you have a good working knowledge of the system. The more you know about the circuit, the easier it will be to isolate the problem. If operating and service manuals are available, familiarize yourself with their contents, so that you know where to go for information if a given problem arises.

The Obvious

Before even turning the equipment on, look for the obvious. Is the power cord plugged-in? Are there any broken or cracked wires, blown fuses, loose connections, loose PC boards, cracked PC boards, circuit discoloration, etc. Any of these very obvious things might lead you quickly to the problem.

After you have given the system a thorough visual inspection, turn it on and operate it. Run it through all of its operating modes in an attempt to duplicate the problem. Sometimes, the equipment fails to operate at all, thereby suggesting a power supply problem. Other times, the equipment operates flawlessly. In this case, you might find after a little investigation that an operator was interpreting a display wrong, or not following the prescribed operating procedures. If there is a legitimate problem, you should be able to duplicate and observe it. Naturally, there are going to be times when the problem is intermittent. Don't jump to conclusions too quickly about a problem's existence.

Once you have identified the problem, use your human senses and commonsense to locate its source before jumping in with all your test equipment. Use your eyes to inspect the entire circuit again, and take your time. Careful examination at this stage might save you a lot of unnecessary work. Again, look for the obvious and, of course, don't forget to look for smoke. Switch off power immediately if it appears.

Use your sense of touch to locate possible problem areas. Overheated components often provide a clue to the problem. The circuit components should be warm or slightly hot to the touch. If it is too hot to touch, it is probably defective or improperly designed, in the case of a prototype. Your sense of smell also gives you a clue to excessive heat problems, since many electronic components give off a unique odor when they overheat. Even your sense of hearing is important, since you might hear the crackling sound of a burning component or an electrical arc.

The most difficult problems are often the simplest and most obvious problems. They are only difficult because they are often overlooked. Over half of the circuit problems you encounter can be isolated and diagnosed at this simple observation stage.

Testing The Power Supply

The power supply should be tested first, since it is common to all the other circuits and is often the source of trouble in a digital system. Check to see if the AC line voltage is present on the input side of the supply. This can be done with a simple multimeter, set on the appropriate (greater than 120-volt) AC voltage scale. If the AC voltage is not present, suspect a fuse or circuit breaker.

Most power supplies are protected with an AC line fuse or circuit breaker. Many times this is the culprit. If you find a blown fuse, unplug the system and replace it with a new one of the same value. If a circuit breaker has tripped, reset it. If the new fuse blows or the circuit breaker trips again, its a good chance that the power supply is defective and should be repaired or replaced.

Assuming the power supply is getting AC power, the next thing to do is check the DC voltages on the output side of the supply. Again, this can be done with a simple multimeter set on the appropriate DC voltage range. Typical circuit voltages are ± 5, ± 12, ± 15, and 24 volts. Make sure these voltages are within their specified limits. If one or more of the DC voltages is not present, the power supply is most likely defective and should be repaired or replaced. The regulator, filter, and rectifier circuits are all common sources of trouble in power supply circuits.

If a DC voltage is lower than it should be, it is probably being loaded too heavily by another circuit. This provides a good opportunity to isolate a problem in a multiboard system down to an individual board. While monitoring the supply voltage with your meter, disconnect one board or circuit module at a time. When the defective board is removed, the supply voltage will return to its normal value. Naturally, you want to switch off system power before removing a unit.

Finally, you might want to look at the DC supply voltages with an oscilloscope. You should see a relatively smooth DC voltage level, with little or no ripple or power supply glitches. If excessive ripple or glitches are observed, suspect the power supply regulator and/or filter circuit.

Testing The Clock Circuit

Aside from the power supply, the clock circuit is one of the most critical parts of the system, since it is common to many of the system ICs. As you know, a sequential circuit will not work without a clock signal.

The first thing you must do to check the clock circuit is to verify the existence of the clock signal. This can be done very easily with a logic probe set in its pulse mode. If the signal is present, the HIGH and LOW LED indicators will both glow dimly, or not at all, and the PULSE indicator will blink. If you don't get this indication, the clock signal is not there and you have isolated the problem to the clock circuit.

Once you have verified the presence of the clock signal with a logic probe, the second thing you should do is to look at it with an oscilloscope. Is it clean and free from noise and glitches? If not, you have found the problem, since digital ICs require a clean noise- and glitch-free clock signal. Next, use the scope to measure the voltage level, frequency, and duty cycle of the clock. Compare your measured values to the clock specifications. If the signal does not meet these specifications, it is a good bet that the trouble is in the clock circuit.

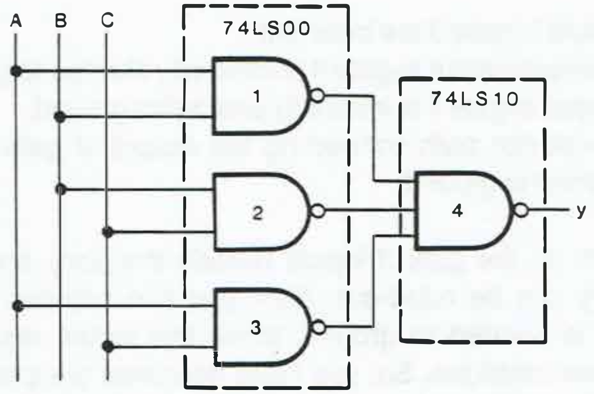
Troubleshooting Combinational Logic

Once you have looked for the obvious, tested the power supply, and tested the clock circuit, the next step is to perform a detailed test of the circuit logic. At this point, you might try substitution.

In a multiboard system, you can substitute replacement boards, one at a time, if they are available. Again, make sure you turn the system power off before unplugging an existing board and plugging-in a replacement board. If a board substitution solves the problem, you have isolated the trouble down to that board. The replacement board gets the system going, and you can troubleshoot and repair the defective board later.

If you suspect a given IC to be defective, you can substitute a replacement IC. Again, make sure the power is off when making the substitution, since removing or installing an IC with the power on can damage the IC. You don't want to create a new problem at this point.

If substitution does not solve the problem, you must perform a detailed test on the circuit logic. Let's use the simple combinational logic circuit in Figure 7-20 as an example. Look familiar? This is the 3-input majority voter circuit that we designed in Unit 2. The first thing you must do when troubleshooting combinational logic is develop a truth table for the circuit being tested so that you know what it is supposed to do. Knowing the operation of the majority voter circuit, you know that you should get a high output whenever two or more of the inputs are high.



ABC	y
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

Figure 7-20

A sample combinational logic circuit, the 3-input majority voter circuit.

To test the circuit, you must apply all the input combinations while monitoring the output with a logic probe or other measuring device. In other words, you must check to see if the circuit is operating according to the truth table. Your goal is to put the circuit in a state that causes an erroneous output, since you cannot troubleshoot a working circuit.

For instance, suppose an input combination of 000 causes the output of the voter circuit to be high. You know this is an erroneous condition from the truth table. Now, what could cause this incorrect output state? If the output from gate 4 were shorted to +V, all of the input combinations would produce a high output. Assuming this is not the case, the next logical step is to check the inputs to gate 4. You know the circuit output should be low for this input combination. So, you ask yourself: Self, what must the inputs to gate 4 be to generate the **correct** low output? Since gate 4 is a NAND gate, the answer is that all the gate inputs must be high. Knowing this, you check each input to gate 4. If all the inputs are high, the gate must be defective and replaced. If one or more of the inputs are low, then the gate is probably working, and you must continue to trace the problem.

Suppose the outputs from gates 1 and 2 are high, but the output from gate 3 is low. The question at this point is: What could cause a low level at this point in the circuit? There are several possibilities:

1. The inputs to gate 3 are incorrect.
2. This particular input to gate 4 is internally shorted to ground.
3. The output of gate 3 is internally shorted to ground.
4. The conductor path connecting the output of gate 3 to the input of gate 4 is shorted to ground.

A quick check on the gate 3 inputs reveals that they are all correct. Thus, the first possibility can be ruled-out. Also, you can rule-out the possibility that the gate 4 input is shorted to ground, since this would result in a logic 1 output for all input combinations. So, you have narrowed the problem down one of two possibilities: the output of gate 3 is internally shorted to ground, or there is a shorted conductor path.

At this point, you should touch the 74LS00. An internal short usually results in an overheated IC. Also, you might want to probe the output of gate 3 with an oscilloscope. Many times, an internal IC short is not a "dead" short, and you will observe some noise riding on the signal. If the problem is due to a dead short of the conductor path, you will not see any noise on the signal.

The last step would to be to remove the 74LS00 from the circuit. Then, with your ohmmeter, make a continuity check between the gate 3 output pin location on the circuit board and circuit ground. If you read a zero resistance, you know that the conductor path is shorted. If this is the case, you must trace the conductor path to locate and repair the short. If you read a high resistance, the IC has an internal short and must be replaced.

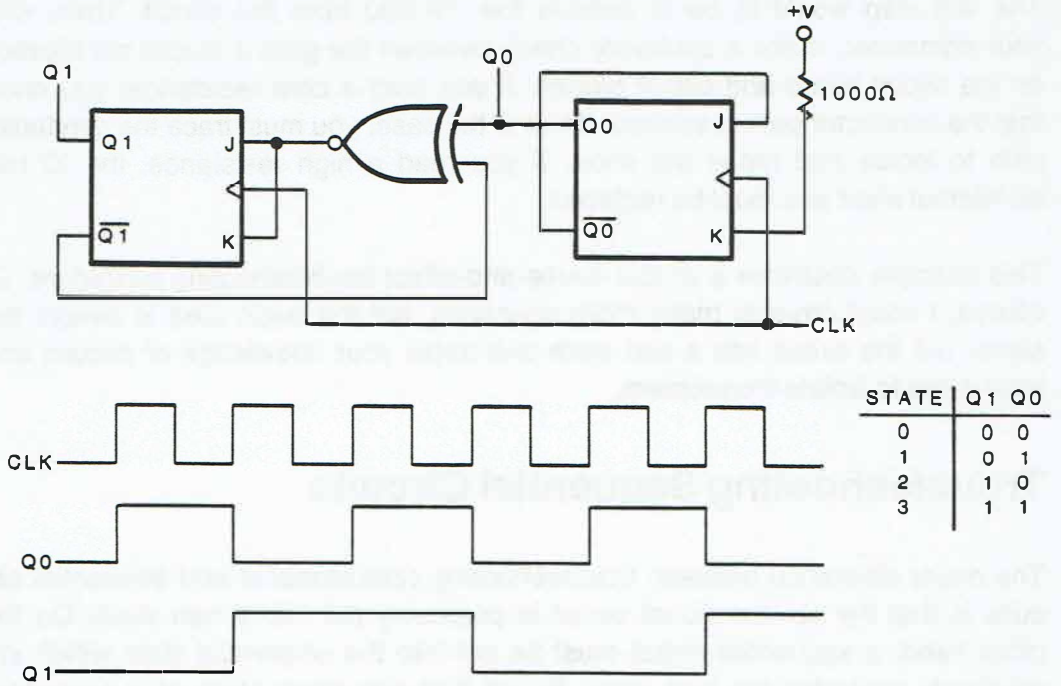
This example illustrates a simple cause-and-effect troubleshooting procedure. Of course, I could develop many more scenarios, but the basic idea is always the same: put the circuit into a bad state and apply your knowledge of circuits and logic gates to isolate the problem.

Troubleshooting Sequential Circuits

The major difference between troubleshooting combinational and sequential circuits is that the combinational circuit is purposely put into a bad state. On the other hand, a sequential circuit must be put into the sequential state which immediately precedes the bad state. Recall that any given state of a sequential circuit is set-up by the previous state. So, the idea is to clock the circuit into a bad state, then return to the previous state so that you can determine why the circuit entered the bad state.

To sequence the circuit, you must disable the system clock and apply a manual clock pulse. This can be done using a simple debounced pushbutton switch. In fact, some commercial circuits include a built-in pushbutton switch which you can use to clock the circuit one pulse at a time. In either case, the manual clock signal allows you to single-step the circuit and observe all the operating states.

Let's use the simple 2-bit counter in Figure 7-21 as an example. The first thing you must do is to determine the circuit state table. From the associated timing diagram, you can see that the circuit should generate the following sequential states: 00, 01, 10, 11, 00, 01, and so on. Suppose that, after clocking the circuit several times, you find the following output sequence: 00, 11, 00, 11, etc. Now, the timing diagram shows that Q0 must toggle on each clock pulse and Q1 must toggle on every other clock pulse. But, the erroneous sequence shows that both flip-flop outputs are toggling on every clock pulse. Thus, the Q0 flip-flop is doing its job, but the Q1 output is the culprit. Since the Q1 flip-flop must be defective, it should be replaced, right? Well, not so fast!



STATE	Q1	Q0
0	0	0
1	0	1
2	1	0
3	1	1

Figure 7-21
A sample sequential circuit.

Notice that the erroneous sequence goes from 00 to 11 when it should go from 00 to 01. So, put the circuit into the 00 state, since this is the state immediately prior to the bad state. Then, using your logic probe, look at the inputs to the Q1 flip-flop. You find them to be at a logic 1 state. So, the flip-flop **should** toggle on the next clock pulse and this is exactly what it is doing. The obvious conclusion is that the Q1 flip-flop is okay.

The next suspect is the 2-bit XNOR gate. What should its inputs be in this state? Following the circuit, the 00 state should produce a 01 input to the gate. The gate has a 01 input, and its producing a logic 1 output. Is this the way an XNOR gate works? Of course not! Your first thought is that the gate is bad, right? This is a possibility. But, the gate output might also be shorted to the +V supply via a shorted conductor path. To determine the exact cause of the problem, you must turn the power off, remove the gate from the circuit, and make a continuity check between the gate output pin location on the circuit board and the +V supply line. You should read infinite resistance if there are no shorted conductor paths. If this is the case, the gate is defective and should be replaced. Otherwise, the conductor path must be traced to locate the short.

Example 7-3

> *Suppose that the circuit in Figure 7-21 generates the following output sequence: 00, 00, 00, 00, etc. What is the likely cause of the trouble?*
>
> Here, the circuit is locked in the 00 state. There are two major defect possibilities: there is no clock signal, or the Q0 flip-flop is defective. Suppose that you check the clock signal and find it to be okay. The next step would be to check the Q0 flip-flop inputs. Since the flip-flop must toggle from 0 to 1 on the next clock pulse, you should find logic 1's on both the J and K inputs. If logic 1's are there, the obvious conclusion is that the flip-flop is defective and must be replaced. Naturally, if one of the logic 1's is missing, you must look for a possible short as I described earlier.

This sequential circuit testing procedure is called **static testing**, since you test the circuit while it is in a stable, or non-changing, state. Another way to troubleshoot sequential circuits is to allow the circuit to run at its normal clock rate. This is called **dynamic testing** and requires the use of an oscilloscope or logic analyzer to monitor the circuit operation. Sometimes, the circuit works fine when you manually single-step it through its logic sequence at a slow rate. However, it doesn't operate properly when it is clocked at the normal high rate. Timing and noise problems often create this symptom. The only way to troubleshoot such a circuit is to allow it to run at normal speed and use an oscilloscope or logic analyzer to isolate the problem.

Repairing the Circuit

This is probably the easiest part of the entire servicing process. If the defective circuit is a new circuit prototype, the appropriate changes must be patched-in on the existing prototype or a new prototype constructed. In any event, don't forget to change the documentation to reflect the circuit alteration. The process of changing an existing design is called **redesign and alter**. The circuit is redesigned and altered to reflect any required changes.

If the defective circuit is an older design and was previously operating properly, the repair might require you to replace a PC board, replace a defective component, or patch a conductor path. Many times, the repair requires you to perform a soldering operation. To replace a defective component, you must unsolder the bad component and solder-in the new one. You will soon learn that unsoldering an IC is a chore. For that reason, it's always a good idea to use IC sockets when you design a prototype circuit.

When you must unsolder and remove a component, be sure to use a solder removing tool to pull the heated solder off the component connections. Such a tool can be a simple vacuum bulb as shown in Figure 7-22a, or a special spring-loaded "solder sucker" that operates on the same principle. Commercial **solder wick** like the one shown in Figure 7-22b is also available to attract the old solder from the heated connection. This is the best method for pulling solder out of a plated-through hole on a double-sided circuit board.

Before removing the defective component, make sure to note the proper component orientation. You surely do not want to install a new IC in the board backwards. When soldering the new component, make sure that you do not overheat the connection. This might cause damage to both the component and the PC board foil. It is always a good idea to heat-sink the component leads while performing the soldering operation. A small pair of needle-nosed pliers or an alligator clip is a good heat-sink for components with long leads, such as resistors, capacitors, diodes, and transistors. Place the pliers or clip on each component lead, before you solder it, between the component and the soldered connection (on the component side of the board). An IC clip will provide some heat-sinking for an IC. Clamp the clip around the IC pins on the component side of the board while you are soldering the pins on the solder side of the board.

Of course, special precautions must be taken when working with MOS devices because they can be damaged by static electricity. These precautions were discussed in Unit 1, but are summarized here for your review:

1. Avoid touching the IC pins, and make sure you are grounded in some fashion.
2. Avoid wearing nylon clothing.
3. Never insert or remove MOS ICs with the power on.
4. When not using a MOS device, store it in conductive foam or an anti-static tube.
5. When soldering the MOS IC pins, make sure the tip of the soldering iron is grounded.

Your final task in the servicing process is to reassemble the system and make sure it is working properly. Run it through its paces several times, trying to repeat the initial failure. It is even a good idea to let it run awhile to "burn-in" any new components. Sometimes, the circuit will work fine initially, then fail after several minutes or hours of operation. This is especially true when overheating problems are not detected and corrected during the servicing operation.

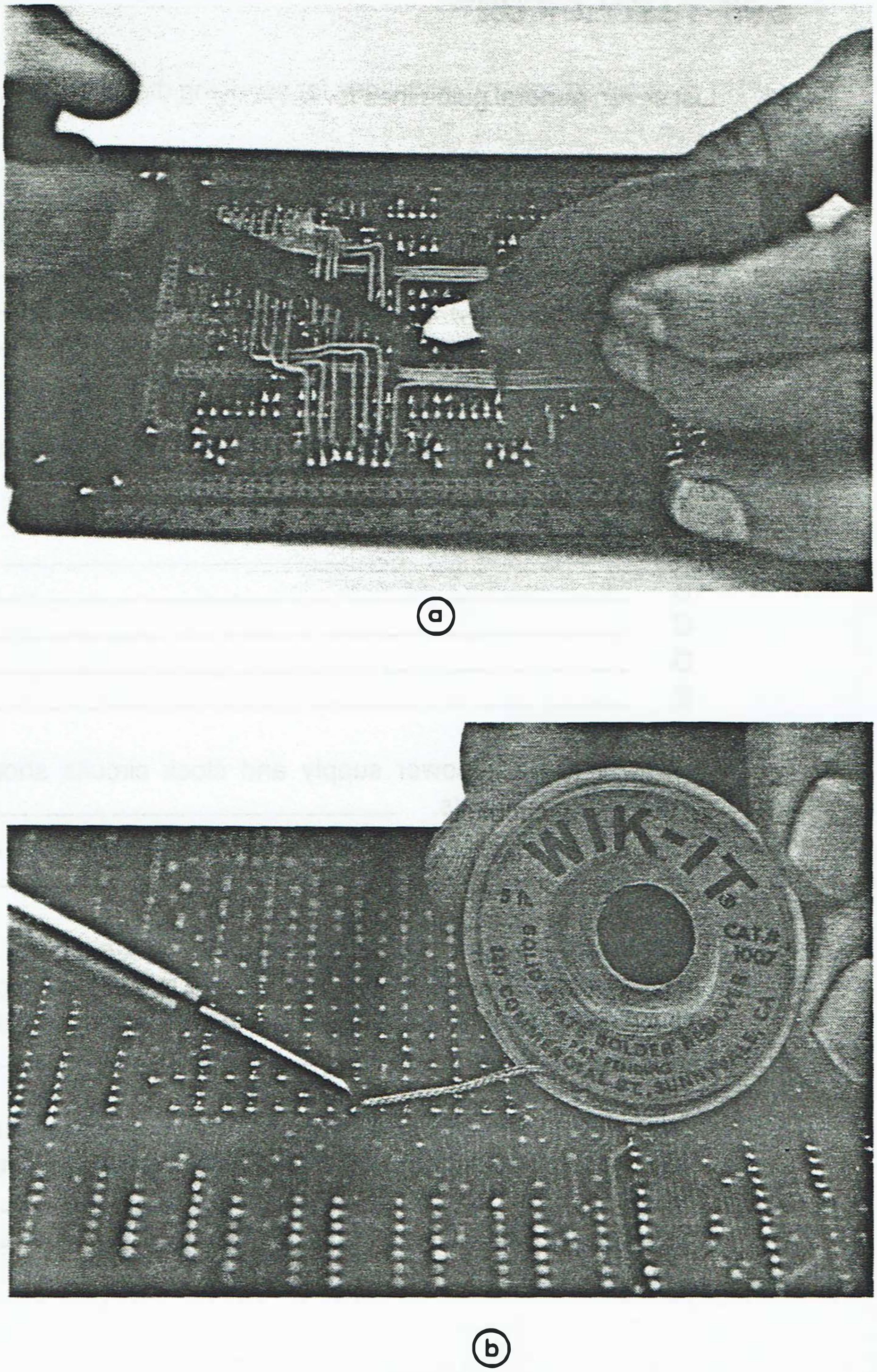


Figure 7-22
Old solder must be removed using a vacuum bulb (a) or solder wick (b) during the unsoldering operation.
(Photgraph, Part (a), by Della Ann Benefiel.)

Self-Test Review

21. List seven general guidelines for servicing digital circuits.

A. ____________________
B. ____________________
C. ____________________
D. ____________________
E. ____________________
F. ____________________
G. ____________________

22. List at least five forms of documentation that are important when servicing digital circuits.

A. ____________________
B. ____________________
C. ____________________
D. ____________________
E. ____________________

23. Explain why the power supply and clock circuits should be tested prior to the other circuits. ____________________

24. When using a logic probe to check the clock signal of a digital circuit the HIGH indicator is off, the LOW indicator illuminates constantly, and the PULSE indicator is blinking. Describe the clock signal. ____________________

25. After testing the power supply and clock circuits, what should be done prior to detailed troubleshooting in multiboard systems? ____________________

26. Suppose you observe a constant logic 1 output from the voter circuit in Figure 7-23 regardless of the input combination. List at least five possible causes of this condition.

A.

B.

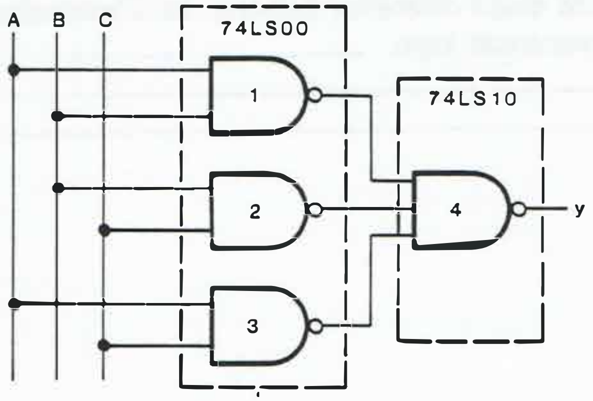
C.

D.

E.

F.

27. What must you do first to begin troubleshooting the problem in question 26?



ABC	y
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

Figure 7-23

Figure for question 26.

28. Suppose you observe the following sequence from the counter circuit in Figure 7-24: 00, 01, 00, 01, 00, etc. List at least five possible causes of this condition.

A. ____________________

B. ____________________

C. ____________________

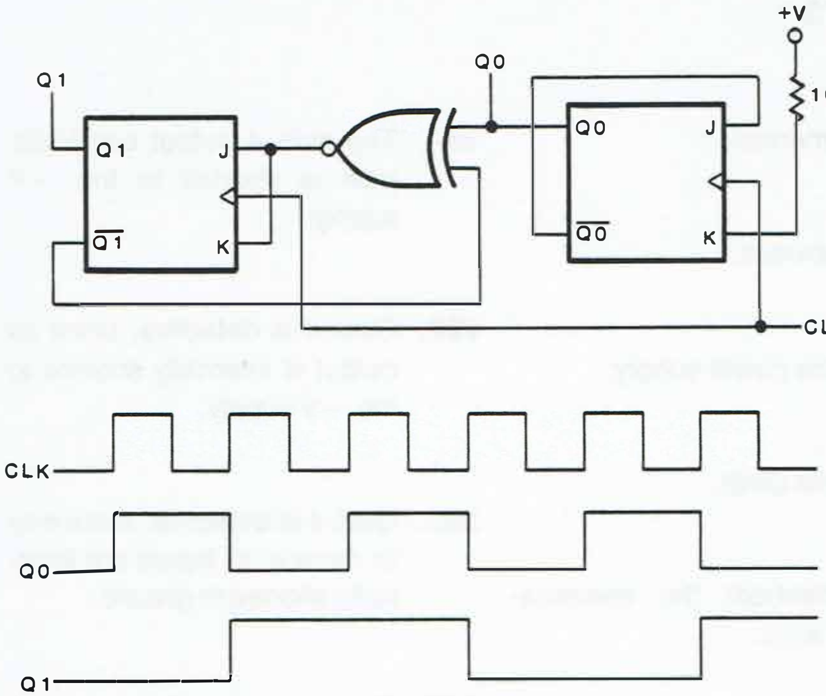
D. ____________________

E. ____________________

F. ____________________

29. What must you do first to begin troubleshooting the problem in question 28? ____________________

30. Explain the major difference between troubleshooting combinational logic versus sequential logic. ____________________



STATE	Q 1	Q 0
0	0	0
1	0	1
2	1	0
3	1	1

Figure 7-24

Figure for question 28.

Answers

21A. Documentation

21B. The obvious.

21C. Test the power supply.

21D. Test the clock.

21E. Troubleshoot the combinational logic.

21F. Troubleshoot the sequential logic.

21G. Repair the circuit.

22. See Table 7-1, Page 7-38.

23. They are common to all the other circuits. Nothing works without a power supply, and sequential circuits do not work without a clock.

24. The clock signal has less than a 20% duty cycle and is most likely defective.

25. You should substitute replacement boards, one at a time.

26A. The gate 4 output conductor path is shorted to the +V supply.

26B. Gate 4 is defective, since its output is internally shorted to the +V supply.

26C. Gate 4 is defective, since one or more of its inputs are internally shorted to ground.

26D. One or more of the gate 4 input conductor paths are shorted to ground.

26E. One or more of the gate 1, 2, or 3 outputs are internally shorted to ground.

26F. The inputs to gate 1, 2, or 3 are incorrect, producing an erroneous output condition.

27. You must first put the circuit in a bad state. Any combination that would not normally produce a logic 1 output is a bad state in this situation.

Answers (continued)

28. With the given output sequence, the Q1 flip-flop is not toggling as it should on every other clock pulse and remains at a constant logic 0 state. Possible causes of this condition include.

    A. the Q1 output conductor path is shorted to ground.

    B. The flip-flop is defective, since its Q1 output is internally shorted to ground.

    C. The flip-flop is defective, since one or both of its JK inputs are internally shorted to ground.

    D. The conductor path between the JK inputs and the XNOR gate output is shorted to ground.

    E. The XNOR gate is defective, since its output is internally shorted to ground.

    F. The clock line to the Q1 flip-flop is open.

29. Put the circuit into a state which immediately precedes the bad state. Here, the circuit goes from 01 to 00, when it should go from 01 ot 10. Thus, the circuit must be put in the 01 state.

30. A combinational circuit is purposely put into a bad state. On the other hand, a sequential circuit must be put into the sequential state that immediately precedes the bad state.

SUMMARY

Design bugs can creep into even the most carefully planned and implemented designs. In combinational logic circuits, common bugs include problems due to noise, power supply glitches, floating inputs, and propagation delay. Sequential designs also suffer from these bugs, but are especially prone to timing problems. Unclocked PRESET and CLEAR inputs, clock skew, and excessive clock rates are all timing problems associated with sequential circuits. In addition, sequential circuits can get locked-up in illegal states if all possible states are not accounted for during the design process.

Test instruments used to troubleshoot digital circuits include the logic probe, logic monitor, logic pulser, multimeter, oscilloscope, and logic analyzer. A logic probe used with a logic pulser provides an economical, efficient, and effective means of troubleshooting many digital circuit problems. Multimeters are used to measure foil continuity, power supply voltages, and logic levels. However, an oscilloscope or logic analyzer must be used if you must actually see the logic signals and perform a dynamic test of the circuit. These instruments allow you to observe signal noise and glitches, along with measuring voltage levels, period, rise time, fall time, and propagation delay time. The logic analyzer is a microprocessor-based instrument that provides three display modes: timing, data, and map.

When troubleshooting a digital circuit, it is important to collect all the documentation available for a given circuit. Then, look for the obvious before even turning the equipment on. The most difficult problems are often the simplest and most obvious problems. They are only difficult because they are often overlooked at this simple observation stage.

Once circuit testing begins, the power supply should be tested first, followed by the clock circuit. These two circuits are tested before detailed circuit troubleshooting begins, since they are common to most of the circuit components and are often a source of trouble in digital circuits.

After the power supply and clock circuits are tested, the next step is to troubleshoot the combinational and sequential logic. For troubleshooting combination logic, you must put the circuit into a bad state, and apply your knowledge of circuits and logic gates to isolate the problem. For static testing of sequential circuits, you must put the circuit into the state immediately prior to the bad state, and apply your knowledge of sequential devices to isolate the problem. Another way to troubleshoot sequential logic is to allow the circuit to run at its normal clock rate and use an oscilloscope or logic analyzer to monitor the circuit operation.

The repair of a new design prototype generally requires a patch circuit, wire-wrap changes, or constructing a new prototype. Don't forget to change the documentation to reflect the design alteration. The repair of an older existing design usually involves unsoldering bad components and soldering new components into the circuit board. Be careful of excessive heat during the soldering operation. You should heat-sink all component leads while they are being soldered. In addition, when unsoldering components, make sure that all the old solder is removed using a vacuum bulb, solder sucker, or solder wick. Special precautions must be taken when replacing MOS components, due to their sensitivity to static electricity.

UNIT EXAMINATION

1. Which of the following is **not** a problem generally associated with combinational logic:
    - A. Noise.
    - B. Clock glitches.
    - C. Power supply glitches.
    - D. Propagation delay.

2. When a TTL input is floating, the logic level seen at the input is:
    - A. Logic 0.
    - B. Logic 1.
    - C. Infinite resistance.
    - D. Unpredictable.

3. The most effective way to prevent power supply glitches from affecting an IC is:
    - A. Use a coupling capacitor between the supply and the IC.
    - B. Shield the IC.
    - C. Use a bypass capacitor on each of the IC input pins.
    - D. Use a bypass capacitor between the +V and ground pins of the IC.

4. To prevent problems due to floating inputs you should:
    - A. Shield the ICs.
    - B. Use bypass capacitors on the ICs.
    - C. Tie the unused logic 1 inputs directly to the +V supply.
    - D. Tie the unused logic 1 inputs to the +V supply via 1000 ohm pull-up resistors.

5. All unused CMOS inputs must be tied to $+V_{DD}$ or V_{SS}.
    - A. True.
    - B. False.

6. A logic signal is to be split in two directions in a digital circuit. One signal path has many more gates than the other. A potential problem exists due to:
    - A. Clock skew.
    - B. Propagation delay.
    - C. Noise.
    - D. Unclocked inputs.

7. Another name for an unclocked input is a(n):
   - A. Asynchronous input.
   - B. Synchronous input.
   - C. Floating input.
   - D. Skewed input.

8. Timing problems usually result when the PRESET and CLEAR inputs are not synchronized with the circuit initialization, or reset, pulse.
   - A. True.
   - B. False.

9. Clock skew occurs due to:
   - A. Propagation delay.
   - B. Floating inputs.
   - C. Unclocked inputs.
   - D. Two or more clock signals being out-of-phase.

10. A clock signal is split in two directions. One signal path is 10 feet longer than the other. The amount of clock skew possible due to wiring delay is approximately:
    - A. 1.5 nS.
    - B. 666 MHz.
    - C. 15 nS.
    - D. 66.6 MHz.

11. The maximum clock frequency for a digital circuit is **not** a function of:
    - A. Flip-Flop delay.
    - B. Gate delay.
    - C. Hold time.
    - D. Set-up time.

12. Hang-up states in sequential circuits result from:
    - A. Unclocked inputs.
    - B. Illegal states.
    - C. Clock skew.
    - D. Excessive clock rates.

13. A logic probe is used to monitor a circuit test point and none of the LED indicators are on. Which of the following is **not** likely?
    - A. The test point is an open circuit.
    - B. The circuit power is not turned on.
    - C. The probe is not connected to power.
    - D. A high frequency (greater than 100 KHz) square wave exists at the test point.

14. A test instrument that performs the opposite function of a logic probe is the:
    - A. Logic Monitor.
    - B. Logic Pulser.
    - C. Logic Clip.
    - D. Logic Analyzer.

15. Which of the following **cannot** be directly measured with an oscilloscope?
    - A. Voltage.
    - B. Frequency.
    - C. Current.
    - D. Propagation delay.

16. When a logic analyzer displays a series of 1's and 0's, it is in its:
    - A. Map mode.
    - B. Data mode.
    - C. Timing mode.
    - D. None of the above.

17. If a power supply voltage output is being loaded too heavily by another circuit it will most likely measure:
    - A. Zero.
    - B. Low.
    - C. High.
    - D. Normal.

18. The inputs to an AND gate are all high, but the output is low. Which of the following is **not** the likely cause of the problem?
    - A. The gate output is internally open.
    - B. The gate output is internally shorted to ground.
    - C. The gate output conductor path is shorted to ground.
    - D. All of the above are likely causes of the problem.

19. A defective 4-bit binary counter exhibits four states that repeat as follows: 0000, 0001, 0010, 1111, 0000, etc. Before beginning to troubleshoot the circuit, it should be clocked into state:
    A. 0000
    B. 0001
    C. 0010
    D. 1111

20. Dynamic testing of a sequential circuit requires the use of a(n):
    A. Multimeter.
    B. Logic probe.
    C. Logic monitor.
    D. Oscilloscope or logic analyzer.

EXAMINATION ANSWERS

1. B — Clock glitches are not a problem generally associated with combinational logic.

2. B — When a TTL input is floating, the logic level seen at the input is a logic 1.

3. D — The most effective way to prevent power supply glitches from affecting an IC is to use a bypass capacitor between the +V and ground pins of the IC.

4. D — To prevent problems due to floating inputs you should tie the unused logic 1 inputs to the +V supply via 1000 ohm pull-up resistors.

5. A — True. All unused CMOS inputs must be tied to $+V_{DD}$ or V_{SS}.

6. B — A logic signal is to be split in two directions in a digital circuit. One signal path has many more gates than the other. A potential problem exists due to propagation delay.

7. A — Another name for an unclocked input is an asynchronous input.

8. B — False. Timing problems usually result when the PRESET and CLEAR inputs are not synchronized with the **clock** signal.

9. D — Clock skew occurs due to two or more clock signals being out-of-phase.

10. C — A clock signal is split in two directions. One signal path is 10 feet longer than the other. The amount of clock skew possible due to wiring delay is approximately 15 nanoseconds (10 × 1.5 nS or 15 nS).

11. C — The maximum clock frequency for a digital circuit is **not** a function of hold time.

12. B — Hang-up states in sequential circuits result from illegal states.

13. D — If a high frequency square wave were at the test point, the PULSE indicator would be blinking.

14. B — A test instrument that performs the opposite function of a logic probe is the logic pulser.

15. C — Current **cannot** be directly measured with an oscilloscope.

16. B — When a logic analyzer displays a series of 1's and 0's, it is in its data mode.

17. B — If a power supply voltage output is being loaded too heavily by another circuit it will most likely measure low.

18. A — If the gate output were internally open, you would not see a low level at this point.

19. C — A defective 4-bit binary counter exhibits four states that repeat as follows: 0000, 0001, 0010, 1111, 0000, etc. Before beginning to troubleshoot the circuit, it should be clocked into state 0010.

20. D — Dynamic testing of a sequential circuit requires the use of an oscilloscope or logic analyzer.

UNIT 8

EXPERIMENTS

CONTENTS

Introduction . . . . Page 8-3

Experiment Format . . . . Page 8-3

Experiment 1: Fundamental Logic Gates . . . . Page 8-5

Experiment 2: Simple Combinational Logic Circuits . . . . Page 8-17

Experiment 3: Data Transmission and Parity . . . . Page 8-25

Experiment 4: Arithmetic Circuits . . . . Page 8-33

Experiment 5: Boolean Algebra . . . . Page 8-39

Experiment 6: Combinational Logic Design Using SSI Devices — Part 1 . . . . Page 8-49

Experiment 7: Combinational Logic Design Using SSI Devices — Part 2 . . . . Page 8-57

Experiment 8: MSI Devices . . . . Page 8-67

Experiment 9: Combinational Logic Design Using MSI Devices . . . . Page 8-75

Experiment 10: Combinational Logic Design Using a ROM . . . . Page 8-85

Experiment 11: Register Circuits . . . . Page 8-93

Experiment 12: Counter Circuits . . . . Page 8-105

Experiment 13: Sequential Circuit Design . . . . Page 8-119

Experiment 14: Display Circuits . . . . Page 8-131

Experiment 15: Digital-to-Analog Conversion . . . . Page 8-137

Experiment 16: Analog-to-Digital Conversion . . . . Page 8-143

INTRODUCTION

The following experiments complement the course text material and allow you to verify many of the concepts presented in the first six units. (There are no experiments for Unit 7 because of the nature of the material presented.) The experiments you will perform have been carefully designed to be useful and enjoyable, but most importantly to demonstrate the learning principles presented in this course. While performing an experiment, don't be a passive bystander — get involved! To really learn from the experiment, you must become involved and continually ask yourself "**Why?** the circuit operates as it does."

You will be using many different ICs in these experiments. All of the pins that must be connected to form a particular circuit are identified in the related circuit diagram. If you wish to determine the function of all the pins in a specific IC, or what the operating characteristics are for that IC, refer to Appendix C of this course. It contains data sheets for each of the ICs.

Experiment Format

The instructions for each experiment are presented in the following format:

Objectives — The objectives state what you will have learned once you complete the experiment. You should keep these objectives in mind as you conduct the experiment.

Introduction — The material under this heading states the purpose of the experiment. It also serves to refresh your memory of the material covered in the corresponding unit in the text.

Material Required — This section provides you with a list of components and other material you will need to complete the experiment. Set these parts aside at the beginning of the experiment, so you will have fewer distractive interruptions (which means fewer wiring errors) during circuit construction.

Procedure — A series of sequential steps which provide the instructions for setting up portions of the experiment. Questions may also be included at different points during the procedure.

Discussion — Each experiment will be discussed to review and highlight the important points you should have observed during the step-by-step procedure. Some of the more lengthy experiments may possibly have more than one discussion, to ensure you have learned a specific point before starting the next procedure.

Grounding Precautions

1. Make sure that you connect all powered equipment to the same power source. This will establish a common reference and prevent ground loops.

2. Make sure that you DO NOT inadvertently insert a non-earth ground into a circuit under test. An improper ground could cause a dangerous situation. You should determine if the reference lead from EACH piece of equipment is earth grounded.

   To make this determination, use an ohmmeter on the R × 1 range to measure the resistance of the reference lead to the center (ground) prong on the equipment's power plug. If the reading is ZERO, then that equipment is earth grounded. With your Trainer and test equipment earth grounded, all measurements must be referenced to ground. We do not recommend that you use any equipment that is not earth grounded.

EXPERIMENT 1

FUNDAMENTAL LOGIC GATES

Objectives:

To investigate typical SSI logic gates.

To construct truth tables of simple logic circuits.

To verify truth tables through experimentation.

To demonstrate the operation of a controlled inverter.

Introduction

In this experiment, you will observe the seven fundamental logical operations invert, AND, OR, NAND, NOR, XOR, and XNOR using standard TTL ICs. You will become familiar with how to wire these ICs and interconnect them to form simple multiple gate circuits.

Before constructing any of the circuits in this experiment, it is important that you know how to insert and remove the digital ICs properly. I suggest you go to Unit 6 at this time and read the short section titled **Solderless Breadboards**, in the main section titled **Digital Circuit Processing**, to learn the techniques for proper IC insertion and removal.

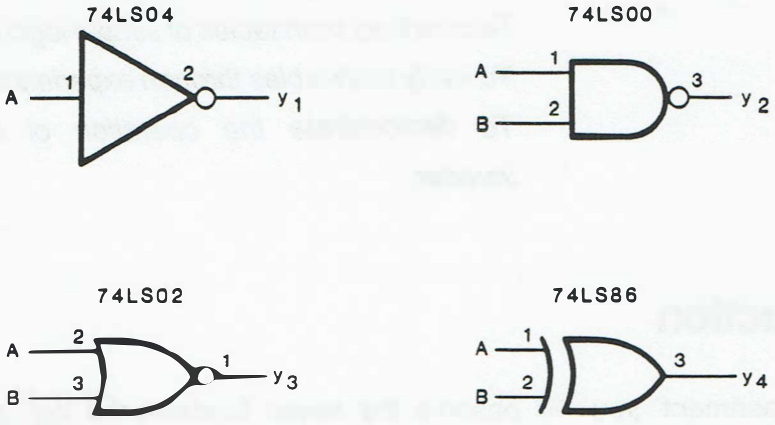
Material Required

Heathkit Engineering Design Trainer ET-1000
1 — 74LS00 Quad 2-bit NAND
1 — 74LS02 Quad 2-bit NOR
1 — 74LS04 Hex Inverter
1 — 74LS10 Triple 3-bit NAND
1 — 74LS86 Quad 2-bit XOR

Procedure

1. Fill-in the truth tables and write the Boolean expressions for each of the logic gate symbols in Figure E1-1.

LOGIC SYMBOLS



TRUTH TABLES

A	y_1
0	
1	

A	B	y_2	y_3	y_4
0	0			
0	1			
1	0			
1	1			

BOOLEAN EXPRESSIONS

y_1 =

y_2 =

y_3 =

y_4 =

Figure E1-1

Simple logic gates.

2. Make sure Trainer power is OFF and wire the 74LS04 inverter to the Trainer by following the wiring diagram in Figure E1-2.

   Orient the IC so that the small notch or hole is on the left as shown. Then, the bottom row of pins are pins 1 through 7 from left to right. The top row of pins are pins 8 through 14 from right to left. These are pointed out in the Inset drawing.

   Notice that the numbers on the inverter symbol in Figure E1-2 are the IC pin numbers that are required to make the inverter connections. The wiring diagram shows how the IC pins must be connected. Here, the input to the inverter is supplied by one of the binary data switches on the Trainer. The inverter output is then connected to an LED indicator on the Trainer. Thus, when wiring the inverter, pin 1 of the 74LS04 must be connected to one of the binary data switches, and pin 2 is connected to one of the logic indicators as shown.

   Don't forget to connect power to the IC. As indicated in the figure, pin 14 must be connected to the +5V supply, and pin 7 must be connected to ground.

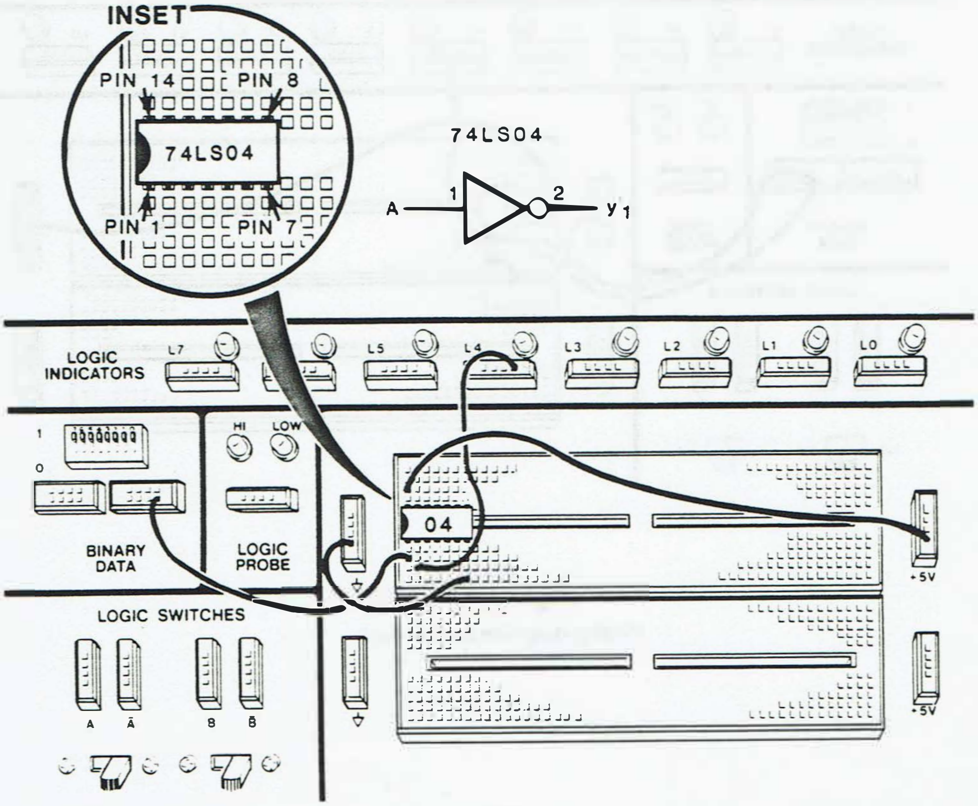


Figure E1-2

Wiring diagram for 74LS04.

3. Switch Trainer power ON and verify the inverter truth table by applying the input logic to the gate. To produce a logic 1 at the gate input, the binary data switch must be in the up position. Conversely, to produce a logic 0, the data switch must be in the down position.

   The LED indicator will light to indicate logic 1 output, and go out to indicate a logic 0 output.

   Does the inverter (gate) produce an inverted output? ______

4. Repeat the process in steps 2 and 3 for the 74LS00 gate in Figure E1-3. Wire a gate to the Trainer as shown, and verify the truth table you developed in Figure E1-1.

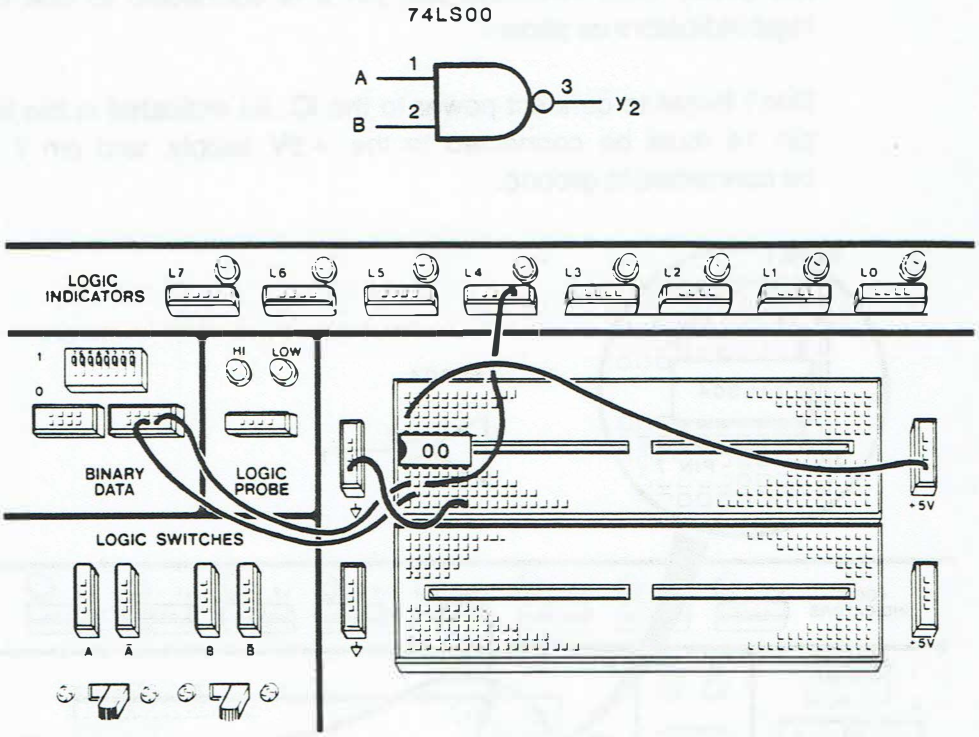


Figure E1-3

Wiring diagram for 74LS00.

5. Again, repeat the process in steps 2 and 3 for the 74LS02 gate in Figure E1-4. Wire a gate to the Trainer as shown, and verify the truth table you developed in Figure E1-1.

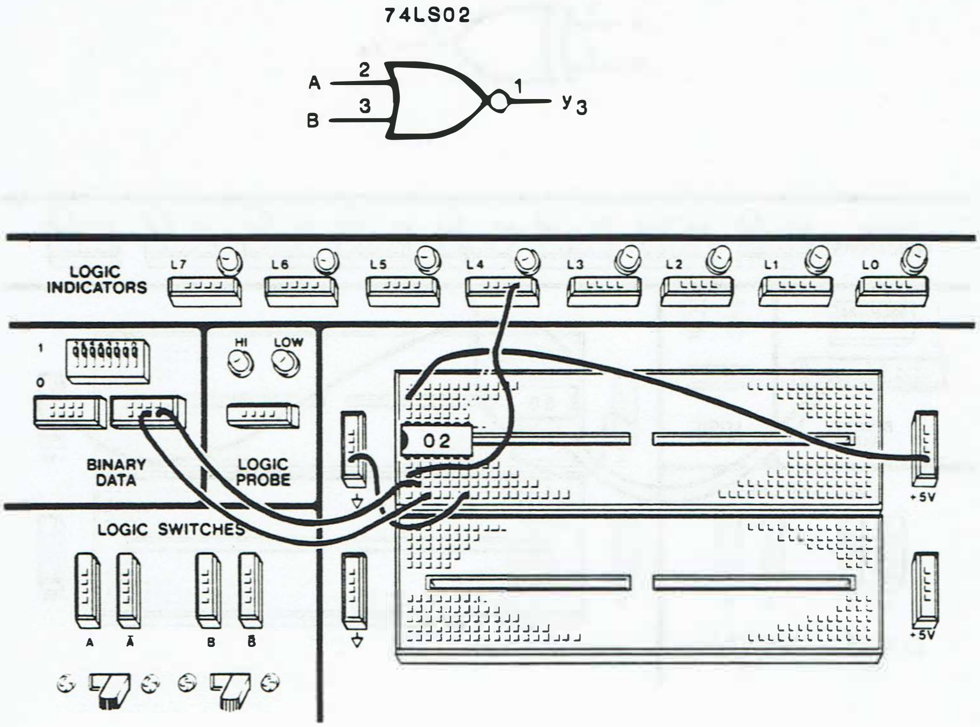
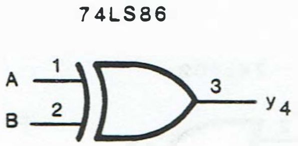


Figure E1-4

Wiring diagram for 74LS02.

6. One last time, repeat the process in steps 2 and 3 for the 74LS86 gate in Figure E1-5. Wire a gate to the Trainer as shown, and verify the truth table you developed in Figure E1-1.



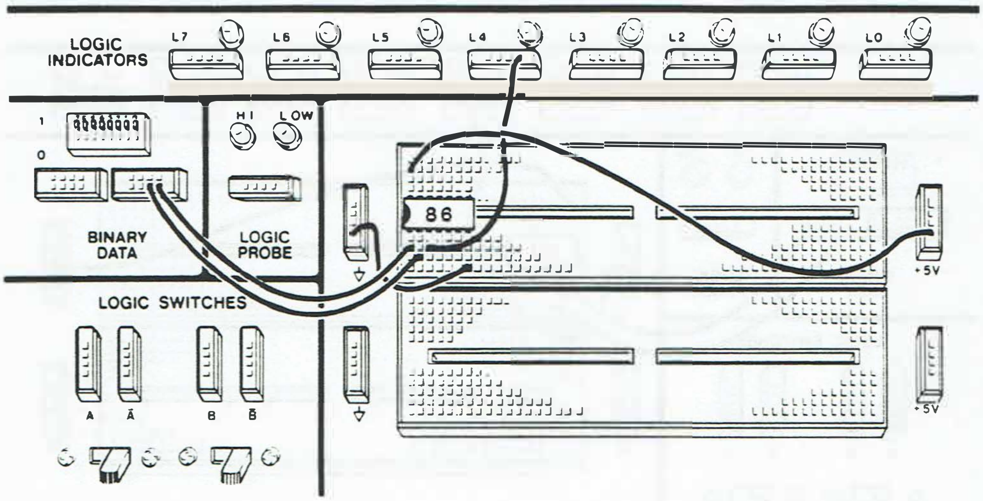


Figure E1-5

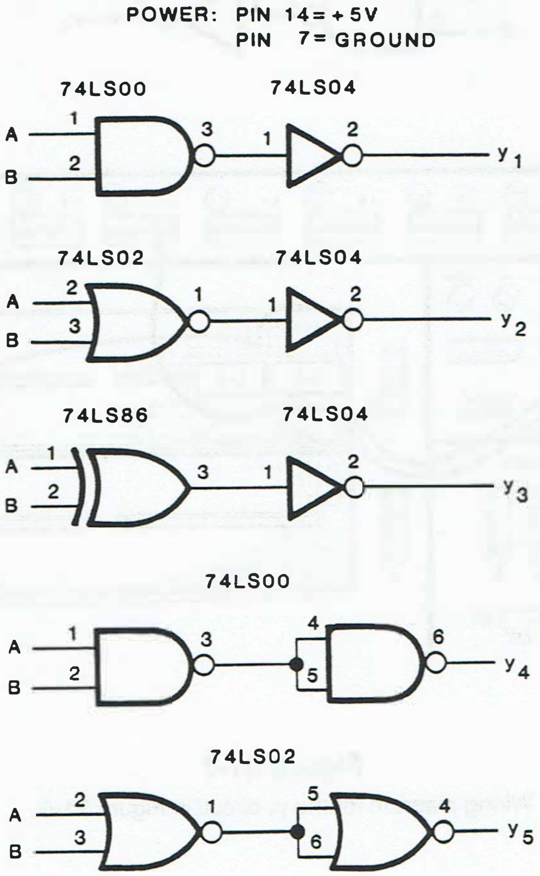
Wiring diagram for 74LS86.

Discussion

The gate outputs should have been identical to their respective truth tables for the various input combinations. Any discrepancy means that either your truth table is wrong, or the gate is not wired properly. This is the beauty of digital electronics: there are no gray areas, either the circuit works as it is supposed to, or something is wrong.

Procedure (continued)

7. Fill-in the truth tables for each of the circuits in Figure E1-6.



TRUTH TABLES

A	B	y_1	y_2	y_3	y_4	y_5
0	0					
0	1					
1	0					
1	1					

Figure E1-6

Some simple logic circuits.

8. Switch Trainer power OFF and construct the first circuit by following the wiring diagram in Figure E1-7.

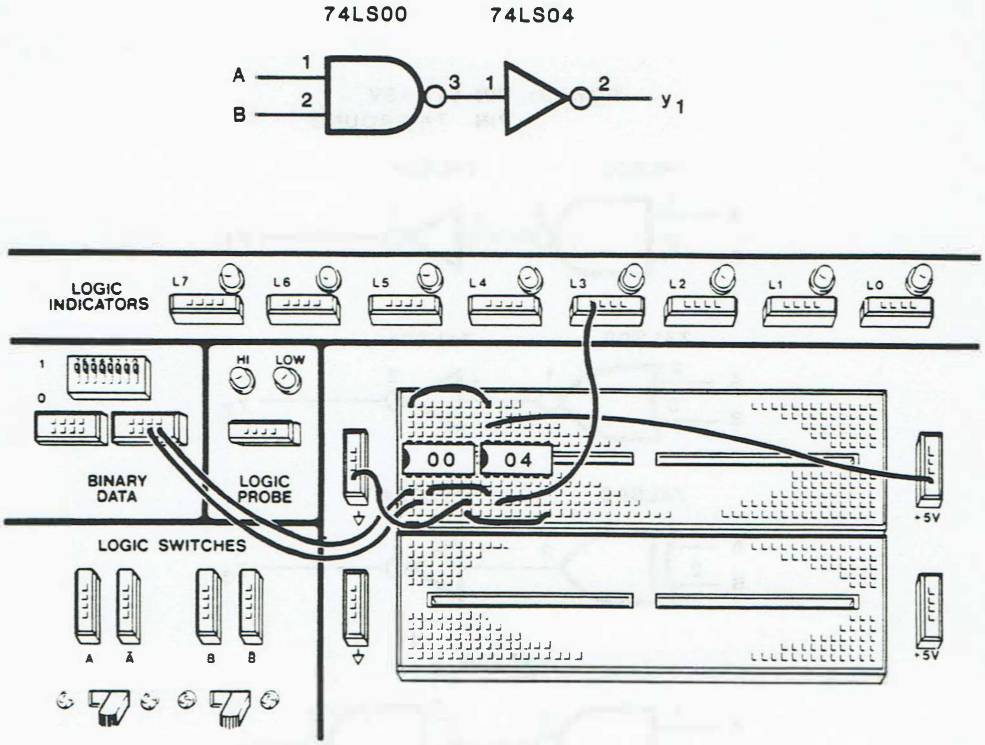


Figure E1-7

Wiring diagram for the y_1 circuit in Figure E1-6.

9. Switch Trainer power ON and verify the truth table for the first circuit by applying all the input combinations via the binary data switches.

What fundamental logic operation does this circuit perform?

10. Repeat steps 6 and 7 for each of the circuits in Figure E1-6. A wiring diagram has not been provided for these circuits. However, you should now be able to wire the circuit by following the IC pin number connections shown in each circuit. For instance, in the second circuit, pins 2 and 3 of the 74LS02 are each connected to a separate binary data switch. Then, pin 1 of the 74LS02 is connected to pin 1 of the 74LS04. Finally, pin 2 of the 74LS04 is connected to one of the Trainer LEDs. Don't forget to connect pin 14 of each IC to +5V and pin 7 of each IC to ground.

    Unless otherwise stated, always use the binary data switches on the Trainer for the circuit inputs and the Trainer LEDs for the circuit output(s).

    I suggest that you make a small checkmark on the circuit diagram for each connection made. This will assure that you have made all the circuit connections, especially when the circuits get more complicated. From now on, all the experiment circuits that you must wire to the Trainer will be shown in schematic form with the required IC pin number connections.

    What fundamental logic operation does each circuit in Figure E1-6 perform? ______________________________

    ____

    What do the last two circuits suggest relative to saving hardware (ICs)? ______________________________

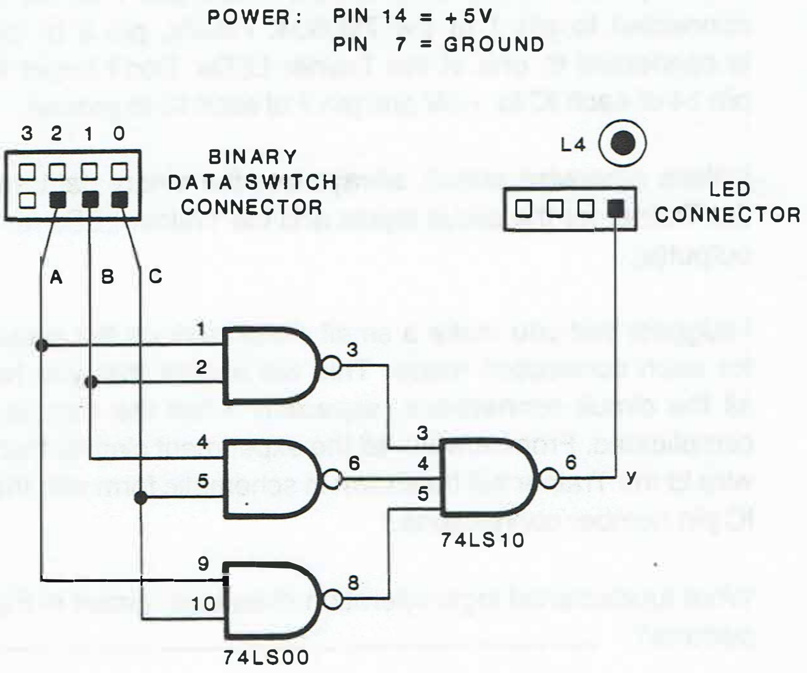
    ____

Discussion

The circuits in Figure E1-6 show you how to produce the AND, OR, and XNOR logic operations using inverters on the outputs of NAND, NOR, and XOR gates, respectively. The last two circuits demonstrate that you can make an inverter by tying the inputs lines of a NAND or NOR together. This allows you to save hardware, since an extra gate in the NAND or NOR IC can be used to form the inverter. Notice that this "little trick" reduces the chip count from 2 to 1 for the AND and OR circuits in Figure E1-6.

Procedure (continued)

11. Switch Trainer power OFF and construct the circuit in Figure E1-8.



A	B	C	y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Figure E1-8

A combinational logic circuit.

12. Switch Trainer power ON and fill-in the associated truth table by applying all the input combinations and observing the actual circuit output.

    What does the circuit do? ______________________

    ____

    Can you suggest a practical application for such a circuit?

    ____

    ____

Discussion

The circuit in Figure E1-8 requires two ICs. The three 2-bit NAND gates are obtained from a single 74LS00 and a 74LS10 supplies the 3-bit NAND gate. Notice that the circuit generates a logic 1 output when two or more of its inputs are high. In other words, the output is high when a **majority** of its inputs are high. Such a circuit might be used as a voter circuit, where the 1 and 0 inputs for A, B, and C, represents the "yes" and "no" votes, respectively, for voters A, B, and C. The output then indicates whether an issue passes with a logic 1 or fails with a logic 0.

Procedure (continued)

13. Switch Trainer power OFF and construct the circuit in Figure E1-9 using a single 74LS86 IC. Use four binary data switches for the four circuit inputs, A_0 through A_3. Use four LED indicators for the four circuit outputs, Y_0 through Y_3. Connect the COMPLEMENT line to the **A** terminal of the spring-loaded logic switch in the lower left-hand corner of the Trainer.

14. Switch power ON and place all the binary data input switches in their logic 0, or down, position and observe the output indicators.

15. Actuate the spring-loaded COMPLEMENT logic switch by moving it to the left, then release it. Observe the output indicators while actuating the switch. What is the circuit doing? ____________

    ____

Discussion

The XOR circuit in Figure E1-9 is called a **controlled inverter**. The operation of the circuit is **controlled** by the COMPLEMENT line. When this line is activated, or logic 1, the circuit inverts, or complements, the A_0 through A_3 input lines.

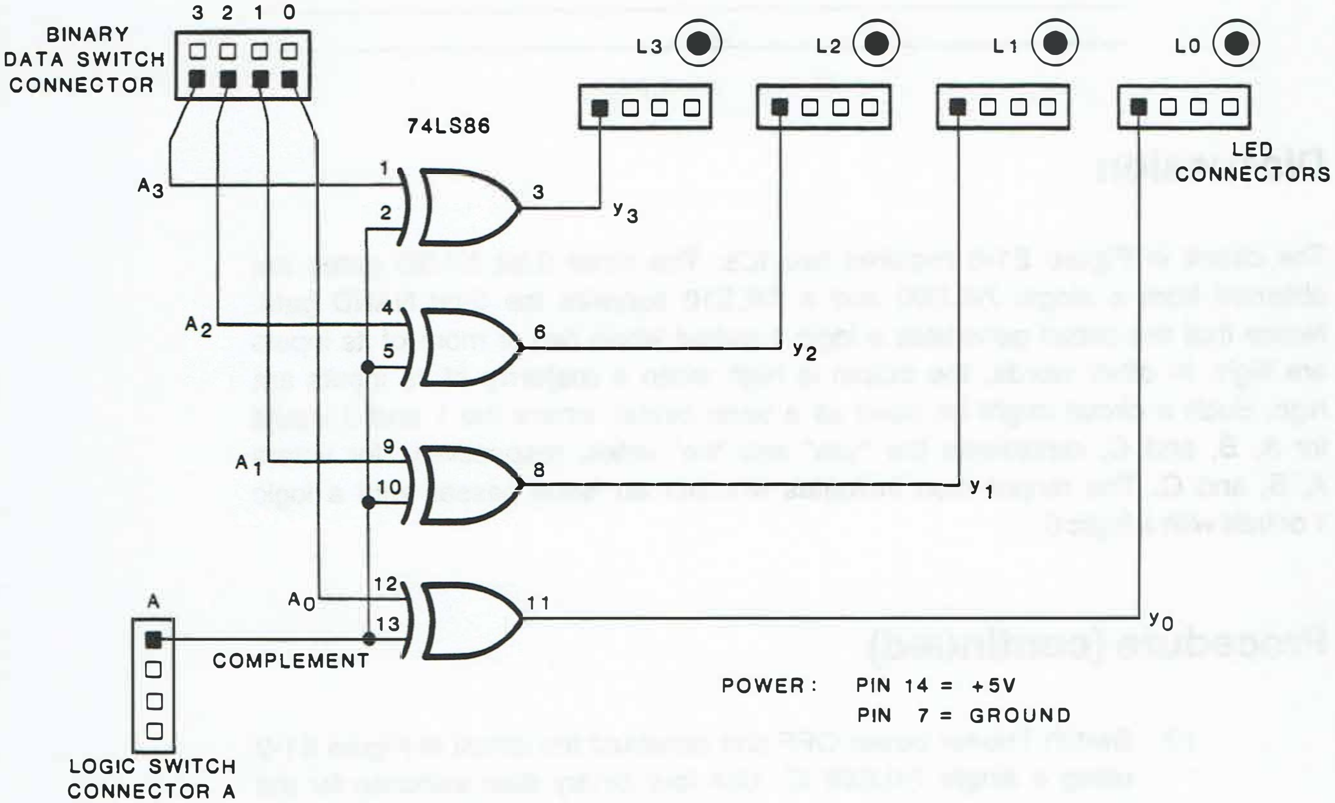


Figure E1-9
A controlled inverter circuit.

Procedure (continued)

16. Switch power OFF and remove the circuit components. Save the ICs and wire for the remaining experiments.

17. Proceed to Experiment 2.

EXPERIMENT 2

SIMPLE COMBINATIONAL LOGIC CIRCUITS

Objectives:

To construct and verify the logic operation of simple logic circuits.

To write Boolean expressions for simple logic circuits.

To demonstrate DeMorgan's theorem experimentally.

To show how to construct 3- and 4-bit gates from 2-bit gates.

Introduction

The most common and least expensive ICs commercially available are NAND, NOR, and XNOR gate ICs. As a result, it is often easier and more economical to implement any combinational logic operation using these three basic gate operations. In fact, many commercial circuits are implemented using only NAND gate ICs, since any logic operation can be performed using various NAND gate combinations.

In this experiment, you will learn how to implement the AND, OR, and XOR operations using NAND and NOR gates. In addition, you will see how 2-bit XOR gates can be cascaded together to implement more than two input variables. You will find all of these little "tricks" useful when implementing your own digital designs in the future.

Remember to make sure the Trainer power is off when constructing or altering the experiment circuits.

Materials Required

Heathkit Engineering Design Trainer ET-1000
1 — 74LS00 Quad 2-bit NAND
1 — 74LS02 Quad 2-bit NOR
1 — 74LS04 Hex Inverter
1 — 74LS86 Quad 2-bit XOR

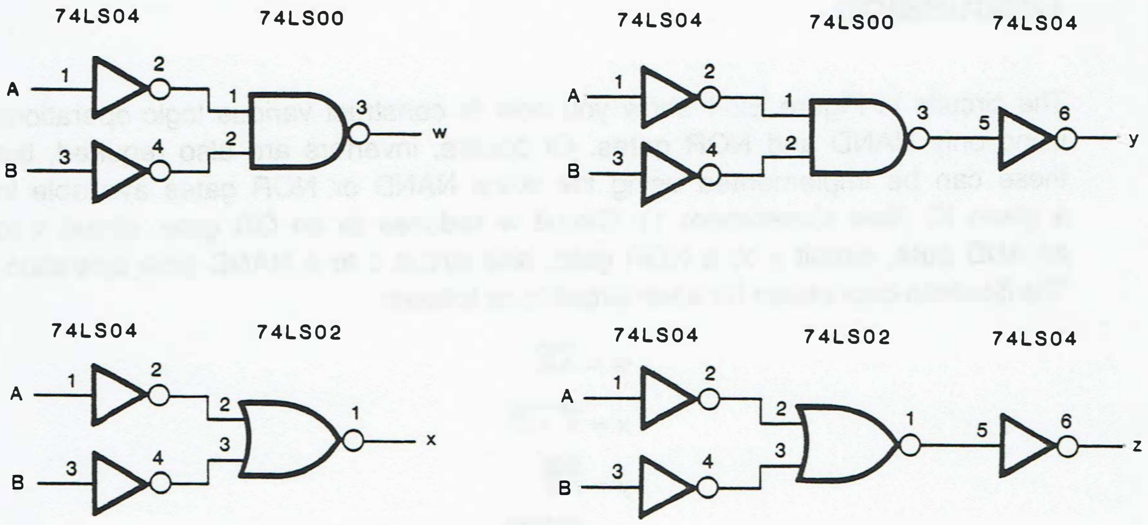
Procedure

1. Write a Boolean expression for each of the circuits shown in Figure E2-1.

2. Fill-in the truth tables for the circuits shown in Figure E2-1.

LOGIC DIAGRAMS

POWER: PIN 14 = +5V
PIN 7 = GROUND



TRUTH TABLES

A	B	w	x	y	z
0	0				
0	1				
1	0				
1	1				

BOOLEAN EXPRESSIONS

w =
x =
y =
z =

Figure E2-1
Gate conversions.

3. Construct each circuit and experimentally verify the truth tables that you developed in step 2. Do not use more than two ICs to implement any given circuit.

   What standard logic operations have been cleverly implemented with each of these circuits? ______________________

   ____

   What useful theorems in Boolean algebra have been demonstrated here? ______________________________

   ____

Discussion

The circuits in Figure E2-1 show you how to construct various logic operations using only NAND and NOR gates. Of course, inverters are also required, but these can be implemented using the extra NAND or NOR gates available in a given IC (See Experiment 1). Circuit w reduces to an OR gate, circuit x to an AND gate, circuit y to a NOR gate, and circuit z to a NAND gate operation. The Boolean expression for each circuit is as follows:

$$w = \overline{\overline{A}\,\overline{B}}$$

$$x = \overline{\overline{A} + \overline{B}}$$

$$y = \overline{\overline{\overline{A}\,\overline{B}}}$$

$$z = \overline{\overline{\overline{A} + \overline{B}}}$$

As a result of your experimentation, you have proven the following Boolean relationships:

$$w = \overline{\overline{A}\ \overline{B}} = A + B$$

$$x = \overline{\overline{A} + \overline{B}} = AB$$

$$y = \overline{\overline{\overline{A}\,\overline{B}}} = \overline{A + B}$$

$$z = \overline{\overline{\overline{A} + \overline{B}}} = \overline{AB}$$

The last two relationships, y and z, are particularly useful for analyzing digital circuits. These two relationships are more commonly known as DeMorgan's 1st and 2nd Theorems, respectively.

Procedure (continued)

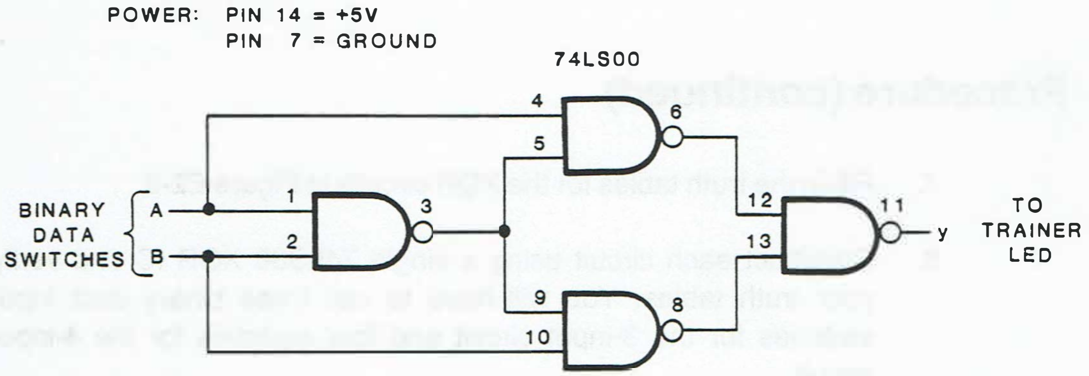
4. Write a Boolean expression for the circuit in Figure E2-2.

5. Fill-in the truth table for the circuit in Figure E2-2.

6. Construct the circuit using a single 74LS00 IC and verify your truth table by experiment.

   What standard logic operation has been implemented by this circuit? ______________________

LOGIC DIAGRAM



TRUTH TABLE

A	B	y
0	0	
0	1	
1	0	
1	1	

BOOLEAN EXPRESSION

y =

Figure E2-2

A NAND gate circuit.

Discussion

The NAND gate circuit in Figure E2-2 performs the operation of a 2-bit XOR gate. The Boolean equation for this circuit is:

$$y = \overline{\overline{[(A)(\overline{AB})]}\,\overline{[(B)(\overline{AB})]}}$$

You have shown experimentally that:

$$y = \overline{\overline{[(A)(\overline{AB})]}\,\overline{[(B)(\overline{AB})]}} = A \oplus B$$

How would you make the preceding circuit perform an XNOR operation? How many ICs would this require? ______________________________

____

____

Now continue with the experiment to verify your answer.

Procedure (continued)

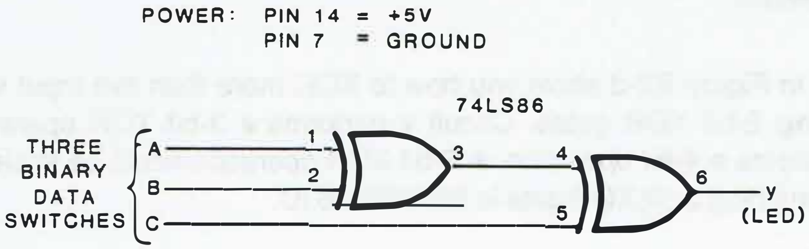
7. Fill-in the truth tables for the XOR circuits in Figure E2-3.
8. Construct each circuit using a single 74LS86 XOR IC and verify your truth tables. You will have to use three binary data input switches for the 3-input circuit and four switches for the 4-input circuit.

   How many input variables have been implemented in each circuit using 2-bit XOR gates? ______________________________

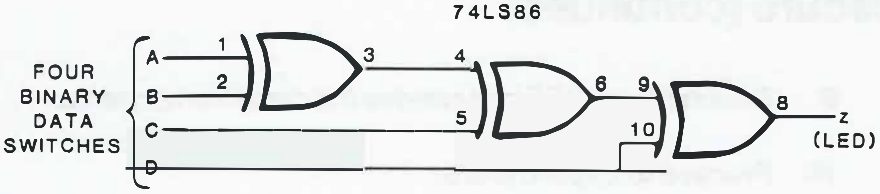
   How could you implement a 5-bit XOR gate using a single 74LS86 Quad 2-bit XOR IC? ______________________________

   How could you make the circuits in Figure E2-3 perform 3- and 4-bit XNOR operations? ______________________________

   Could XNOR gates be "cascaded" in the same way to XNOR more than two input variables? ______________________________



A	B	C	y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



A	B	C	D	z
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Figure E2-3

XOR gate circuits.

Discussion

The circuits in Figure E2-3 show you how to XOR more than two input variables by **cascading** 2-bit XOR gates. Circuit y performs a 3-bit XOR operation and circuit z performs a 4-bit operation. A 5-bit XOR operation could be implemented using the remaining 2-bit XOR gate in the 74LS86 IC.

A 3- or 4-bit XNOR operation could be implemented by adding an inverter to the respective circuit outputs in Figure E2-3. Of course, this requires another IC.

Although you can cascade smaller XOR gates to implement more input variables, you cannot achieve the same result with XNOR gates. In other words, cascading two 2-bit XNOR gates will **not** produce a 3-bit XNOR operation. In fact, if you cascade two XNOR gates in this manner you get a 3-bit XOR operation. Try it!

Procedure (continued)

9. Switch power OFF and remove the circuit components.
10. Proceed to Experiment 3.

EXPERIMENT 3

DATA TRANSMISSION AND PARITY

Objectives:

To simulate a data transmission system that generates and checks parity.

To design both even and odd parity data transmission systems.

To demonstrate the use of XOR and XNOR gates to generate and detect parity in data transmission systems.

Introduction

The block diagram of data a transmission system is shown in Figure E3-1a. Let's first suppose that it is an even parity system and works as follows: The system transmits a 4-bit word consisting of three data bits A, B, and C plus a parity bit, P. The transmitter generates the three data bits, which are monitored by the parity bit generator. If the parity of the three data bits is odd, the parity bit generator sets the parity bit, P, to a logic 1 to make the 4-bit word (ABCP) even parity. Conversely, if the parity of the data bits is even, the parity bit generator clears the parity bit to a logic 0 to maintain the even parity. The result being that the the 4-bit transmitted word is **always even parity**. Although the system is an even parity system, the parity bit generator must recognize odd parity. Why? What common gate detects odd parity?

The 4-bit even parity word is transmitted to the system receiver, where it is monitored by a parity detector. The job of the parity detector is to provide an error indication if it detects an odd parity word in the system. Since an even parity system is assumed, an odd parity word represents and error condition. What common gate recognizes odd parity?

We'll answer these questions in this experiment.

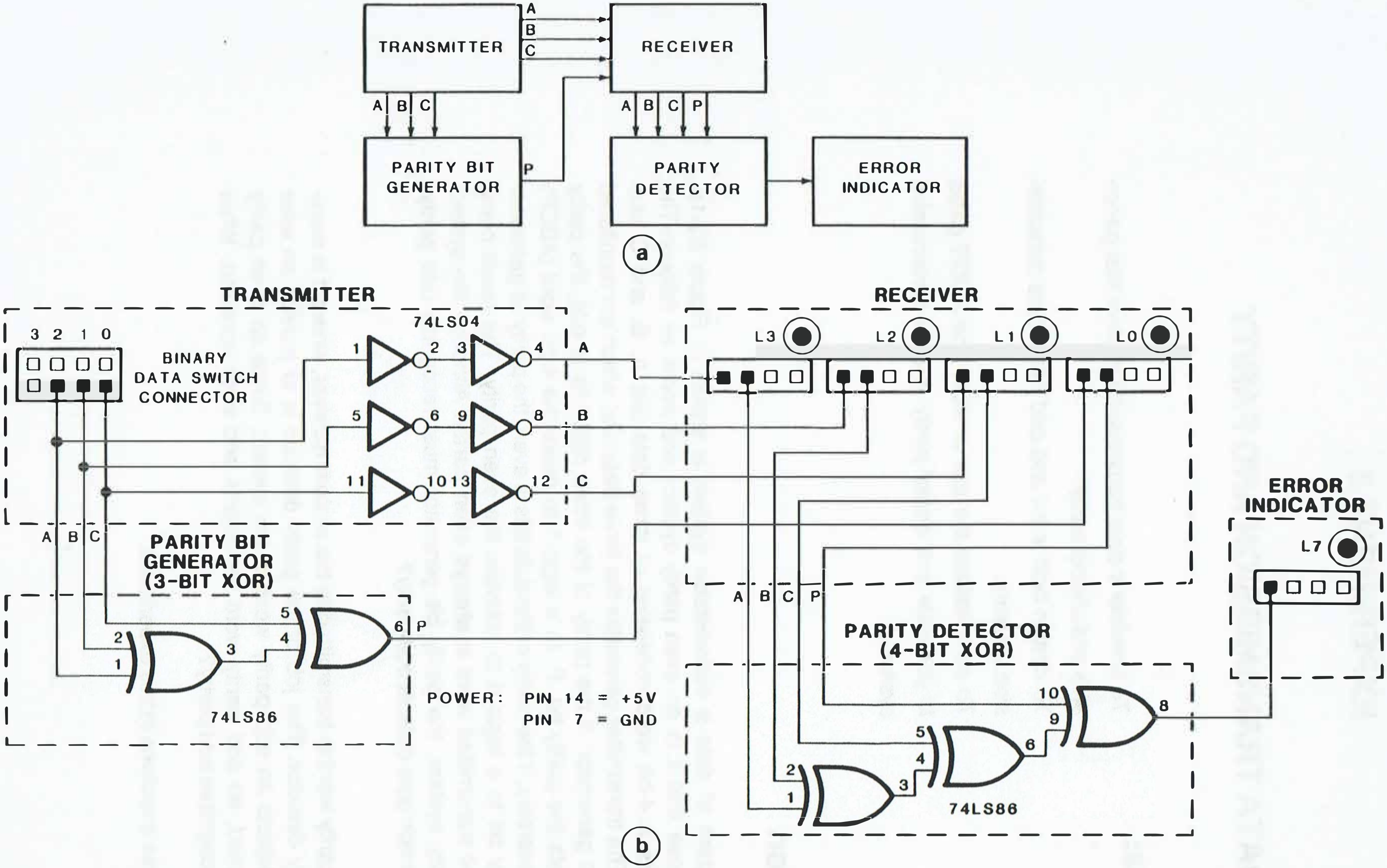


Figure E3-1

A parity system: block diagram (a) and circuit (b).

Materials Required

Heathkit Engineering Design Trainer ET-1000
2 — 74LS04 Hex Inverter
2 — 74LS86 Quad 2-bit XOR

Procedure

1. Construct the data transmission system in Figure E3-1b. Use the following components for each of the system black-boxes:

   Transmitter: Three binary data switches to generate the ABC logic. Six hex inverters to provide the proper buffering and logic levels between the switches and the LEDs.

   Even Parity Bit Generator: A 3-bit XOR gate.

   Receiver: Four LED indicators to show the 4-bit system word.

   Odd Bit Detector: A 4-bit XOR gate.

   Error Indicator: A single LED indicator.

   Note: You must construct the 3- and 4-bit XOR gates from the 2-bit gates available in the 74LS86 ICs (see Experiment 2). Use one 74LS86 IC for the parity bit generator and the other 74LS86 IC to make the parity detector.

2. Exercise the circuit by providing all possible ABC input combinations using the three binary data switches. Observe the receiver LEDs. If the circuit is working properly, you should always see even parity displayed by the LEDs and never get an error indication.

3. To simulate a transmission line error, switch power OFF and place an inverter in series with any one of the data transmission lines (A, B, or C), between the transmitter and receiver as shown in Figure E3-2. Use the first inverter in the second 74LS04 inverter IC package.

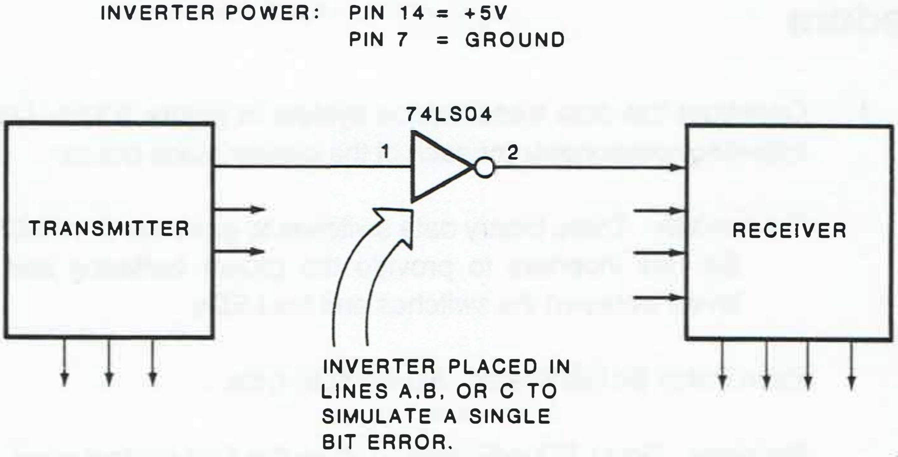

INVERTER POWER: PIN 14 = +5V
PIN 7 = GROUND
74LS04
1
2
TRANSMITTER
RECEIVER
INVERTER PLACED IN LINES A,B, OR C TO SIMULATE A SINGLE BIT ERROR.

Figure E3-2

Simulating a single-bit error in a parity system.

4. Switch power ON and observe the LED indicators. For any input switch combination, you should see odd parity on the LED indicators. In addition, the error indicator should illuminate to show the error condition.

Discussion

Except for the hex inverters in the transmitter, the circuit operation was explained in the introduction. The inverters are used to isolate the LEDs from the switches. Without the isolation, the LED drive requirements will pull the +5-volt logic level from the switches down to about 1.6 volts. Recall that this is a point midway between a logic 1 and a logic 0. The first inverter in each data line provides the isolation, while the second inverter reinverts the logic to its original level.

Although the system is an even parity system, an XOR gate must be used for both the parity generator and the parity detector. This is necessary, since in both cases, odd parity must be recognized by the gate structure.

Parity systems are used in computers when data must be transmitted over long distances. The transmitted word is always one parity, even or odd. If a bit is changed during the transmission, it changes the parity. Such bit-errors are usually caused by electrical noise from motors, generators, and atmospheric disturbances. When a receiving device receives the transmitted code, it must check to see if the word is the correct parity. If not, an error indication must be generated to show the error condition.

Procedure (continued)

5. Switch power OFF and insert another inverter in one of the other data lines to simulate two bit-errors as shown in Figure E3-3.

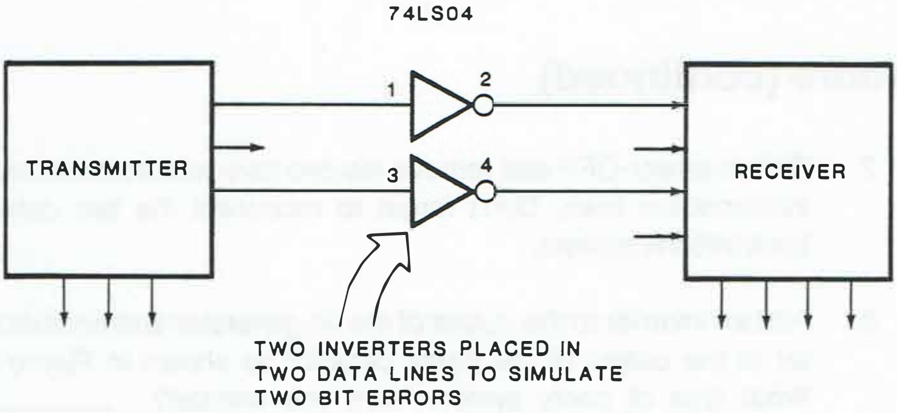

74LS04
1
2
3
4
TRANSMITTER
RECEIVER
TWO INVERTERS PLACED IN TWO DATA LINES TO SIMULATE TWO BIT ERRORS

Figure E3-3

Simulating two bit-errors in a parity system.

6. Switch power ON an generate several input combinations via the binary data switches. Observe the LED indicators.

What parity is the transmitted word? ____________________

Is there an error condition? ____________________

Discussion

By inserting another inverter into another data line, you have simulated two data bit-errors. However, no error indication was generated. Does this mean that the circuit cannot detect more that one bit-error? Yes, and no. The circuit will detect when an odd number of bits are wrong, but not when an even number of bit-errors exist. Thus, two inverted data bits go undetected, but one or three inverted bits will be detected. Why?

The possibility that more than one bit will be erroneous in any given data transmission word is remote. For this reason, most parity systems are only designed to check for single bit errors.

How can the even parity system be converted to an odd parity system?

Procedure (continued)

7. Switch power OFF and remove the two inverters from the two data transmission lines. Don't forget to reconnect the two data lines back into the system.

8. Add an inverter to the output of the bit generator and another inverter to the output of the parity detector as shown in Figure E3-4. What type of parity system have you created? ____________

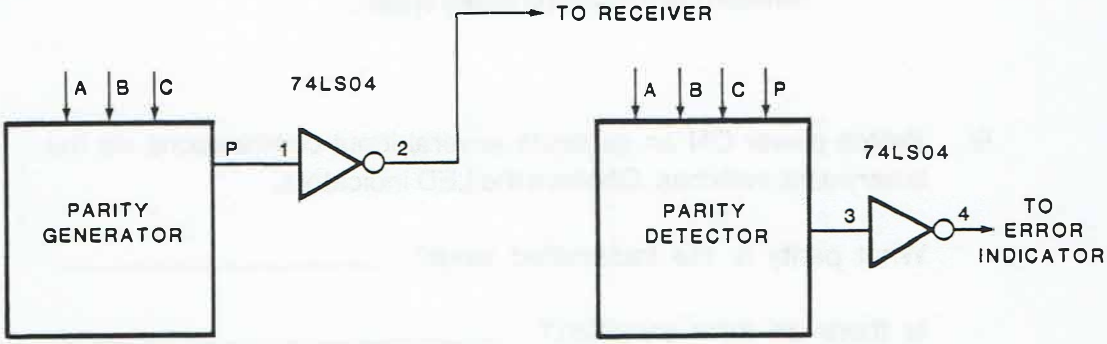


Figure E3-4

Converting an even parity system to an odd parity system.

9. Switch power ON and exercise the circuit by generating several data input combinations using the binary data switches. Observe the LED indicators.

   You should see that the transmitted word is always odd parity and there are no parity errors. Thus, you have created an odd parity system.

10. Switch power OFF and place an inverter in series with one of the data lines to simulate a single bit-error.

11. Switch power ON and observe the LED indicators as you exercise the circuit. The transmitted word should now be even parity for any input combination as shown by the receiver LEDs. In addition, the error indicator should be illuminated to show an error condition for this odd parity system.

Discussion

By adding inverters to the output of the parity generator and detector, you have converted the XOR gates to XNOR gates. This, in turn, makes the system an odd parity system. In summary, you can say that XOR gates are required to generate and detect the parity in an even parity system, while XNOR gate must be used to generate and detect parity in an odd parity system.

12. Switch power OFF and remove the circuit components and wire.

13. Proceed to Experiment 4.

EXPERIMENT 4

ARITHMETIC CIRCUITS

Objectives:

To investigate some basic arithmetic circuits.

To construct circuits that will add two and three binary bits.

To develop a design theory for circuits that will add two n-bit binary numbers.

To explain how digital computers perform the four basic arithmetic operations of addition, subtraction, multiplication, and division.

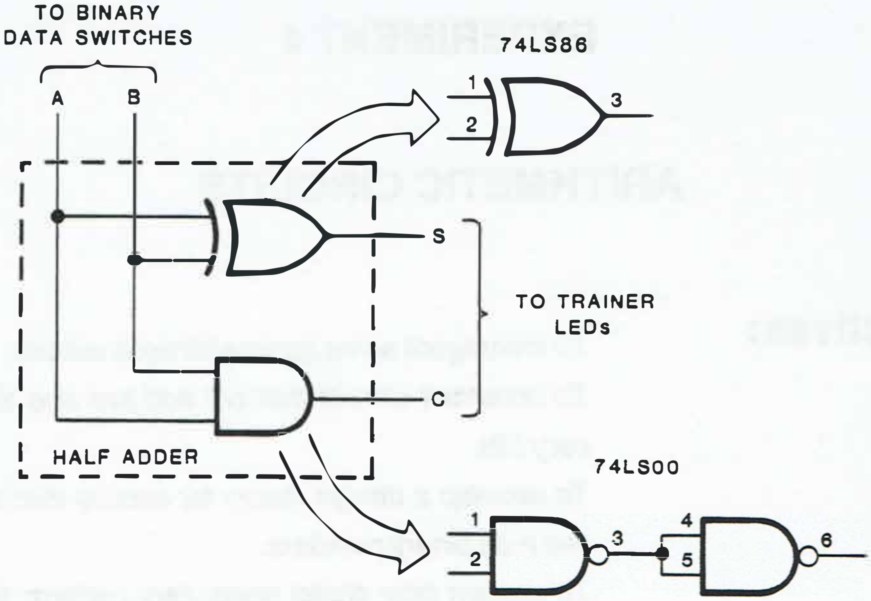
Introduction

Aside from making decisions, combinational logic circuits are also used to perform arithmetic operations. The part of a digital computer that performs all arithmetic operations is called the **arithmetic logic unit**, or **ALU**. The ALU is basically an addition circuit that consists of a **half adder** and several **full adder** circuits. A half adder adds two bits, while a full adder adds three bits.

In this experiment, you will construct both half and full adder circuits. Then, you will learn how to connect them together to create addition circuits that can be used to add two binary numbers of any size. You will also learn how the other three arithmetic operations of subtraction, multiplication, and division are derived from addition.

Materials Required

Heathkit Engineering Design Trainer ET-1000
2 — 74LS00 Quad 2-bit NAND
1 — 74LS02 Quad 2-bit NOR
2 — 74LS86 Quad XOR



A	B	S	C
0	0		
0	1		
1	0		
1	1		

Figure E4-1
A half-adder circuit.

Procedure

1. Determine the truth table for the circuit in Figure E4-1.

2. Construct the circuit, then switch power ON and verify your truth table by experiment. You must use a 74LS00 NAND IC to construct the AND gate as shown. A 74LS86 is used to provide the XOR gate.

   Notice that the trainer breadboard consists of two breadboard blocks, one above the other. Construct this circuit on the upper breadboard block.

   What is the significance of the S and C outputs as related to binary addition?

   ____

   ____

Discussion

The circuit in Figure E4-1 is called a **half adder**. By definition, a half adder is a circuit that adds two binary bits. You should have observed that the S output is the **S**um of the two input bits, while C is the **C**arry output. Notice that if you add 1 and 1 in binary, you get a sum of 0 with a carry of 1. (Refer to Appendix A if you don't understand this concept.)

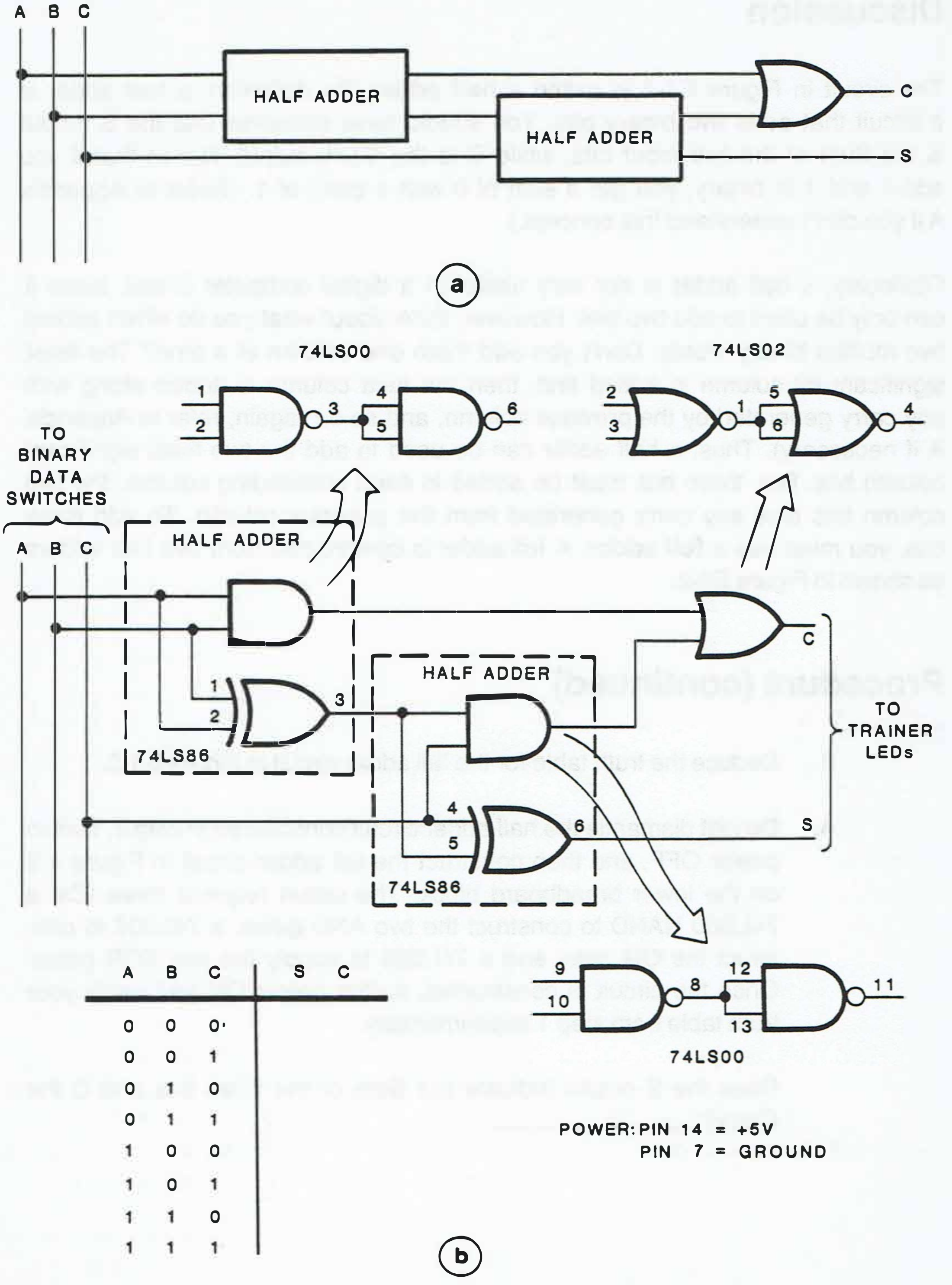
Obviously, a half adder is not very useful in a digital computer circuit, since it can only be used to add two bits. However, think about what you do when adding two multibit binary words. Don't you add them one column at a time? The least significant bit column is added first, then the next column is added along with any carry generated by the previous column, and so on (again, refer to Appendix A if necessary). Thus, a half adder can be used to add the two least significant column bits. But, three bits must be added in each succeeding column: the two column bits plus any carry generated from the previous column. To add three bits, you must use a **full adder**. A full adder is constructed from two half adders as shown in Figure E4-2.

Procedure (continued)

3. Deduce the truth table for the full adder circuit in Figure E4-2.

4. **Do not** dismantle the half adder circuit constructed in step 2. Switch power OFF, and then construct the full adder circuit in Figure 4-2 on the lower breadboard block. The circuit requires three ICs: a 74LS00 NAND to construct the two AND gates, a 74LS02 to construct the OR gate, and a 74LS86 to supply the two XOR gates. Once the circuit is constructed, switch power ON and verify your truth table from step 1 experimentally.

   Does the S output indicate the **S**um of the three bits and C the **C**arry? _______________



A	B	C	S	C
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Figure E4-2

A full adder circuit.

Discussion

A full adder adds three binary bits and is constructed from two half adders. In Figure E4-2, the first half adder adds bits A and B to produce a sum which is added to C by the second half adder. A carry is generated if either of the half adders produce a carry.

To add two n-bit numbers, you must use a half adder to add the least significant bit column and full adders to add each succeeding bit column. Why?

A block diagram of an adding circuit that adds two 2-bit binary numbers is shown in Figure E4-3. Here, the two numbers to be added are **A** = A1 A0 and **B** = B1 B0. The circuit output is the binary sum, **S** = S2 S1 S0. Notice that the most significant sum bit, S2, is the final, or last, carry bit.

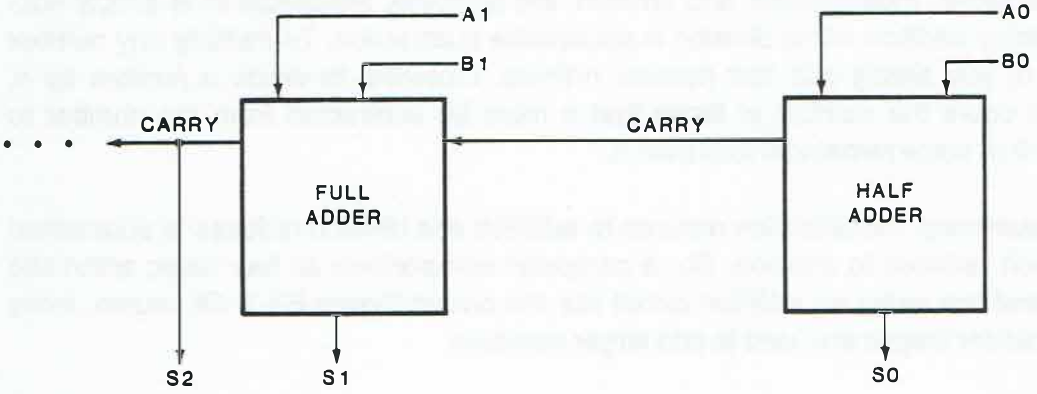


Figure E4-3

Adding two 2-bit binary numbers.

Procedure (continued)

5. Switch power OFF and construct the adding circuit in Figure E4-3. Use four binary data switches, two for A1 A0 and two for B1 B0. Use three LED indicators to show the sum, S2 S1 S0. Use the half and full adder circuits from Figure E4-1 and E4-2, respectively, for the blocks in Figure E4-3.

6. Switch power ON and verify that the circuit adds the two 2-bit input words. For example, if **A** = 10 and **B** = 11 you should get a sum of: **S** = 101.

   How many half adders and full adders would be required to add two n-bit numbers? ______________________________

Discussion

You have just learned how digital computers are designed to add. In fact, a computer can perform any arithmetic operation using addition. From algebra, you know that subtraction is simply a special case of addition, since you actually **add** a negative to subtract. For instance, to subtract 3 from 5, you take the negative of 3 and add it to 5. In symbols: $5 - 3 = 5 + (-3) = 2$.

This is precisely the way a digital computer subtracts. Now the other two arithmetic operations, multiplication and division, are possible. Multiplication is simply successive addition, while division is successive subtraction. To multiply any number by n, you simply add that number n-times. Likewise, to divide a number by n, you count the number of times that n must be subtracted from the number to get 0 or some remainder less than n.

In summary, multiplication reduces to addition and division reduces to subtraction which reduces to addition. So, a computer can perform all four basic arithmetic operations using an addition circuit like the one in Figure E4-3. Of course, more full adder stages are used to add larger numbers.

Procedure (continued)

7. Switch power OFF, and remove and save the circuit components and wire.

8. This completes the experiments for Unit 1. Return to the unit and take the Unit Examination.

EXPERIMENT 5

BOOLEAN ALGEBRA

Objectives:

To write Boolean expressions for different points in a circuit.
To sketch circuits from Boolean expressions.
To simplify digital circuits using Boolean algebra.
To convert AND/OR circuits to NAND/NOR circuits.
To prove Boolean relationships experimentally.
To use a logic probe to trace the logic in a circuit.

Introduction

Boolean algebra is important in digital circuit design because it allows you to represent digital circuits mathematically. By applying the laws of Boolean algebra, logic expressions can be reduced, resulting in simpler and more economical circuits. Few areas in science exhibit such a direct relationship between mathematics and the real world.

Table E5-1 lists several of the more useful laws of Boolean algebra. In this experiment, you will write Boolean expressions for logic circuits then use the laws in Table E5-1 to simplify the expressions. You will then implement the simplified expression to provide the same logic operation as the original circuit.

Table E5-1: Some Common Laws of Boolean Algebra

1.	$\overline{\overline{A}} = A$	8.	$A \cdot A = A$
2.	$A + 1 = 1$	9.	$(A+B) \cdot (A+C) = (A+B)(A+C) = A+BC$
3.	$A \cdot 1 = A$	10.	$A \cdot (B+C) = A(B+C) = AB+AC$
4.	$A + A = A$	11.	$A + \overline{A}B = A + B$
5.	$A + \overline{A} = 1$	12.	$\overline{A+B} = \overline{A}\,\overline{B}$ (DeMorgan's 1st)
6.	$A \cdot \overline{A} = 0$	13.	$\overline{AB} = \overline{A} + \overline{B}$ (DeMorgan's 2nd)
7.	$A \cdot 0 = 0$	14.	$A\overline{B} + \overline{A}B = A \oplus B$

Materials Required

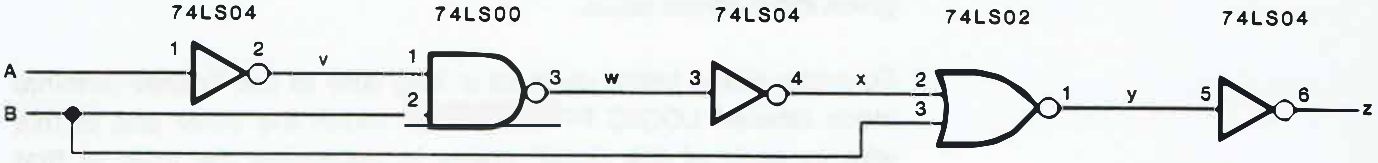
Heathkit Engineering Design Trainer ET-1000
1 — 74LS00 Quad 2-bit NAND
2 — 74LS04 Hex Inverter
1 — 74LS02 Quad 2-bit NOR

Procedure

1. Write a Boolean expression which logically describes each of the five labeled points in the circuit in Figure E5-1.

   Can any of the expressions be simplified using the laws of Boolean algebra? _______________

POWER : PIN 14 = +5V
PIN 7 = GROUND



TRUTH TABLES

BOOLEAN EXPRESSIONS

v =
w =
x =
y =
z =

A	B	v	w	x	y	z
0	0					
0	1					
1	0					
1	1					

Figure E5-1

Write a Boolean expression for points v, w, x, y, and z.
Deduce the truth tables from the expressions.

2. Deduce the truth tables for each of the points labeled in Figure E5-1. Use your Boolean expressions from step 1 for this operation.

For example, the equation at point w is:

$$w = \overline{\overline{A}B}$$

Substituting the AB input logic combinations you get:

$$w = \overline{\overline{A}B} = \overline{\overline{0}0} = \overline{10} = 1$$
$$w = \overline{\overline{A}B} = \overline{\overline{0}1} = \overline{11} = 0$$
$$w = \overline{\overline{A}B} = \overline{\overline{1}0} = \overline{00} = 1$$
$$w = \overline{\overline{A}B} = \overline{\overline{1}1} = \overline{01} = 1$$

3. Construct the circuit, then switch power ON and verify each of the truth table values you obtained in step 2. Use the logic probe on the trainer to determine the logic at each circuit point for a given input combination.

   To make these tests, connect a long wire to the Trainer terminal block labeled LOGIC PROBE. Then touch the other end of this wire to each of the circuit points to determine the logic at that point. A red indication means a logic 1, while a green indication means a logic 0 exists at the point.

   Begin these tests by applying an input combination of AB = 00, then probe each of the labeled points in the circuit to determine the logic state at that point. Then continue with the three remaining input combinations.

Discussion

Boolean algebra is used to describe digital circuits. The Boolean expressions for the labeled points in Figure E5-1 are:

$$v = \overline{A}$$

$$w = \overline{\overline{A}B}$$

$$x = \overline{\overline{\overline{A}B}} = \overline{A}B$$

$$y = \overline{\overline{\overline{\overline{A}B}}+B} = \overline{\overline{A}B+B} = \overline{B(\overline{A}+1)} = \overline{B(1)} = \overline{B}$$

$$z = \overline{\overline{\overline{\overline{\overline{A}B}}+B}} = \overline{A}B+B = B(\overline{A}+1) = B(1) = B$$

Notice that equations x, y, and z, can be simplified using the laws of Boolean algebra. In fact equation y reduces to $\overline{B}$, and equation z reduces to simply B. This demonstrates the power of Boolean algebra for simplifying digital circuits.

Procedure (continued)

4. Sketch the AND/OR circuits that will implement both sides of Boolean Law 9.

5. Convert your AND/OR designs to NAND/NOR designs using the gate conversion chart in Figure E5-2 on Page 8-44.

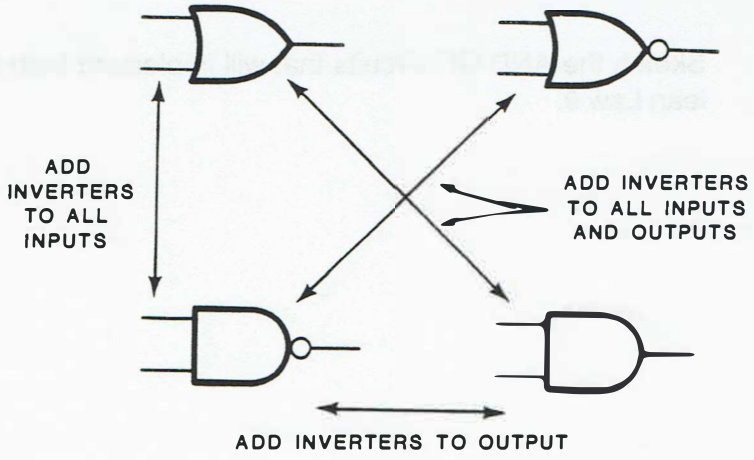

ADD INVERTERS TO ALL INPUTS
ADD INVERTERS TO ALL INPUTS AND OUTPUTS
ADD INVERTERS TO OUTPUT

Figure E5-2

The gate conversion chart.

6. Switch power OFF and construct each circuit from your NAND/NOR designs in step 5. Then switch power ON and verify that the two circuits generate identical truth tables.

Discussion

Boolean Law 9 states that:

$$(A+B)(A+C) = A+BC$$

If you let

$$y_1 = (A+B)(A+C)$$

and

$$y_2 = A+BC$$

the AND/OR circuits you designed for both y_1 and y_2 should match those shown in Figure E5-3. Using the gate conversion chart, you should get the respective NAND/NOR circuits in Figure E5-4. Here, y_1 has been converted to a NAND design and y_2 has been converted to a NOR design. Of course, other NAND/NOR combinations are possible and your conversions might not be the same, even though they are correct.

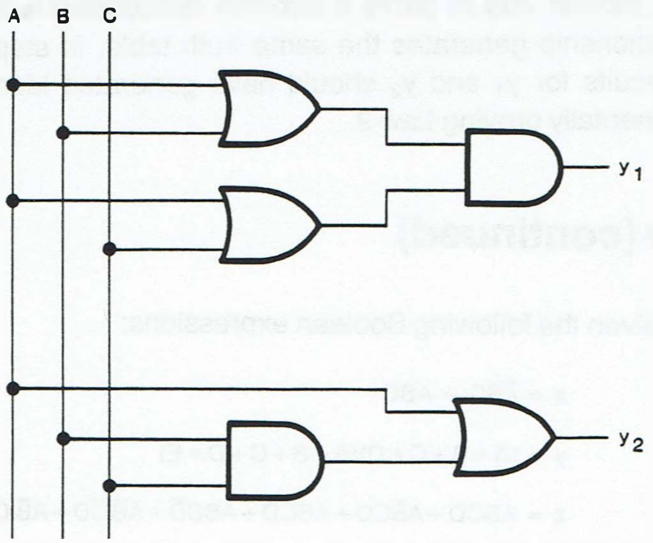


Figure E5-3

AND/OR circuits for Boolean Law 9.

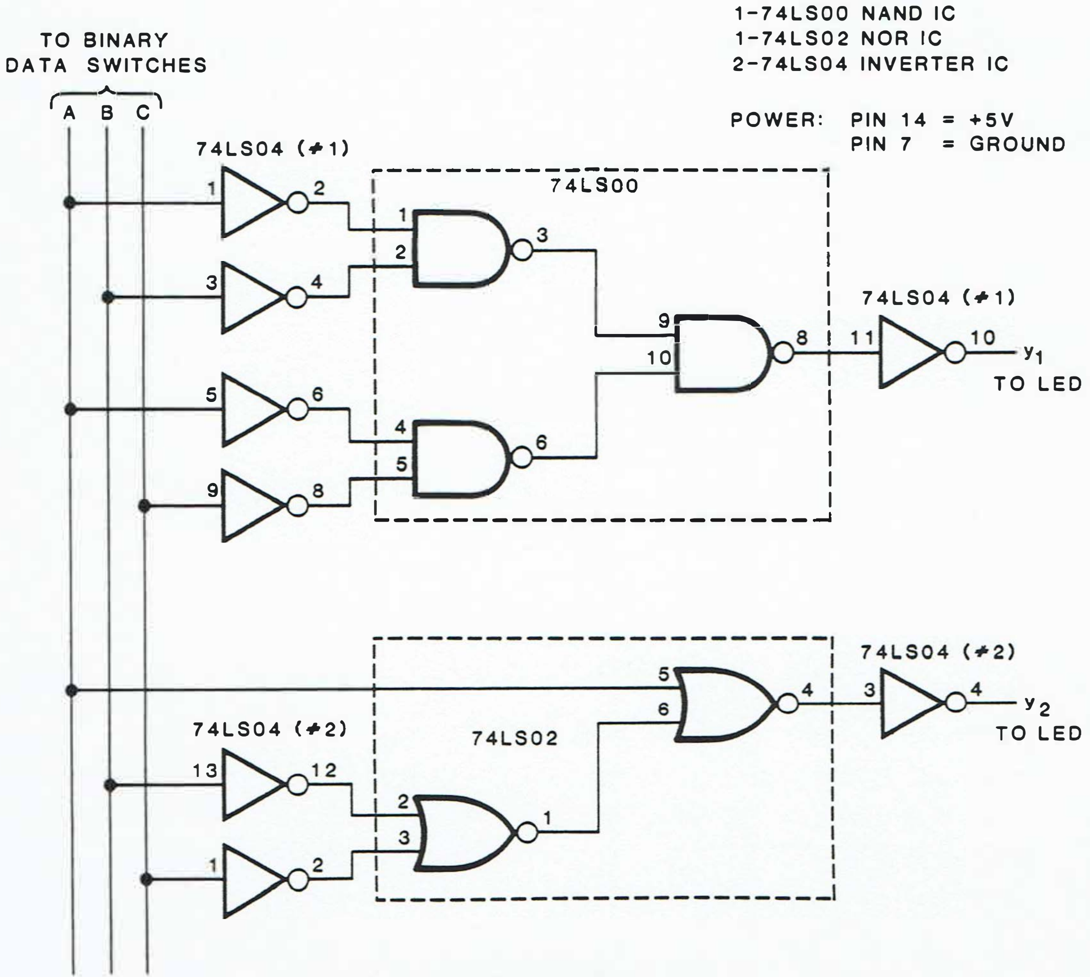


Figure E5-4

NAND/NOR circuits for Figure E5-3.

Recall that the easiest way to prove a Boolean relationship is to prove that each side of the relationship generates the same truth table. In step 6 your separate NAND/NOR circuits for y_1 and y_2 should have generated identical truth tables, thereby experimentally proving Law 9.

Procedure (continued)

7. Given the following Boolean expressions:

$$x = \overline{A}BC + ABC$$

$$y = (A+B+C+D)(A+B+C+D+E)$$

$$z = ABCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D} + A\overline{B}\,\overline{C}D + A\overline{B}\,C\overline{D} + AB\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}\,\overline{D}$$

   Simplify each of the expressions using the laws of Boolean algebra.

8. Sketch the AND/OR circuits from the simplified expressions obtained in step 7.

9. Convert your AND/OR designs to NAND/NOR designs using the gate conversion chart.

10. Switch power OFF and implement your NAND/NOR designs on the Trainer. Then switch power ON and determine the respective truth tables experimentally.

Discussion

Boolean algebra is used to represent and simplify digital circuits. By simplifying the Boolean expression, you simplify the logic required to implement the circuit. This reduces the required chip count and results in a more economical design.

The Boolean expressions given in step 7 are simplified using the Laws of Boolean algebra as follows:

Expression x:

$$\begin{aligned} x &= \bar{A}BC + ABC \\ &= BC(\bar{A} + A) && \text{Law 10} \\ &= BC(1) && \text{Law 5} \\ &= BC && \text{Law 3} \end{aligned}$$

Expression y:

$$y = (A + B + C + D)(A + B + C + D + E)$$

To reduce the confusion of such a complex expression, let $k = (A+B+C+D)$. Then by substitution:

$$
\begin{aligned}
y &= k(k+E) && \text{Substitution}\\
&= kk+kE && \text{Law 10}\\
&= k+kE && \text{Law 8}\\
&= k(1+E) && \text{Law 10}\\
&= k(1) && \text{Law 2}\\
&= k && \text{Law 3}\\
&= (A+B+C+D) && \text{Substitution}
\end{aligned}
$$

Expression z:

$$
\begin{aligned}
z &= ABCD+A\overline{B}CD+AB\overline{C}D+ABC\overline{D}+A\overline{B}\,\overline{C}D+A\overline{B}C\overline{D}+AB\overline{C}\,\overline{D}+A\overline{B}\,\overline{C}\,\overline{D}\\
&= A(BCD+\overline{B}CD+B\overline{C}D+BC\overline{D}+\overline{B}\,\overline{C}D+\overline{B}C\overline{D}+B\overline{C}\,\overline{D}+\overline{B}\,\overline{C}\,\overline{D}) && \text{Law 10}\\
&= A(BCD+B\overline{C}D+\overline{B}CD+\overline{B}\,\overline{C}D+BC\overline{D}+B\overline{C}\,\overline{D}+\overline{B}C\overline{D}+\overline{B}\,\overline{C}\,\overline{D}) && \text{Rearrange}\\
&= A[BD(C+\overline{C})+\overline{B}D(C+\overline{C})+B\overline{D}(C+\overline{C})+\overline{B}\,\overline{D}(C+\overline{C})] && \text{Law 10}\\
&= A[BD(1)+\overline{B}D(1)+B\overline{D}(1)+\overline{B}\,\overline{D}(1)] && \text{Law 5}\\
&= A(BD+\overline{B}D+B\overline{D}+\overline{B}\,\overline{D}) && \text{Law 3}\\
&= A[D(B+\overline{B})+\overline{D}(B+\overline{B})] && \text{Law 10}\\
&= A[D(1)+\overline{D}(1)] && \text{Law 5}\\
&= A(D+\overline{D}) && \text{Law 3}\\
&= A(1) && \text{Law 5}\\
&= A && \text{Law 3}
\end{aligned}
$$

As you can see, the simplified expressions are very easy to implement. Expression x reduces to a single 2-bit AND gate, while y reduces to a 4-bit OR gate. Expression z looks complicated in the beginning, but with the power of Boolean algebra it reduces to a single wire connection to the A input.

Procedure (continued)

11. Switch power OFF and remove the circuit components and wire.
12. Return to Unit 2 and read the section “SSI Implementation.”

EXPERIMENT 6

COMBINATIONAL LOGIC DESIGN USING SSI DEVICES — PART 1

Objectives:

To apply the SSI design algorithm to combinational logic design problems.

To implement your design and verify that it meets the application task.

Introduction

In Unit 2, you learned how to design combinational logic circuits using the SSI design algorithm. Recall the following design steps:

I/O Definition — Define all inputs and outputs from the application task.

Logic Definition — Define the meaning of a logic 0 and a logic 1 for each input and output.

Truth Table — Develop a truth table from the application task and your I/O and logic definitions.

Boolean Expression — Write a Boolean expression for each output in the truth table.

Simplification — Simplify each Boolean expression using the laws of Boolean algebra or Karnaugh mapping.

AND/OR Design — Sketch an AND/OR circuit from the simplified expression.

NAND/NOR/XOR Design — Convert the AND/OR design to a NAND/NOR/XOR design and implement the final design.

Given an application task, the preceding design algorithm provides you a means of designing the simplest logic circuit that will satisfy the design requirements. In this and the next experiment, you will apply the design algorithm to design combinational logic circuits that will satisfy several real-world control tasks.

Materials Required

Heathkit Engineering Design Trainer ET-1000
1 — 74LS00 Quad 2-bit NAND
1 — 74LS04 Hex Inverter
1 — 74LS86 Quad 2-bit XOR

Procedure

1. Given the following application problem:

   An industrial process is to be controlled by means of three binary metallic pressure switches. Each switch is to be a snap-action type switch that changes state relative to a preset pressure level. In other words, each switch will remain in one position until the pressure reaches a specific point and causes the switch to "snap" into the other position. When the pressure drops below the "trip" level, the switch snaps back to its original condition, or state. There are two actuators controlled by the switches: a fuel supply valve and an electrical pump.

   Now suppose the switches are labeled S1, S2, and S3 and the fuel valve and pump must be controlled as follows:

   A. If all three switches are activated by a rise in pressure, the fuel shuts off and the pump goes on.

   B. If only S1 and S3 are activated, fuel is supplied and the pump goes on.

   C. If only S1 and S2 are activated, both the fuel and pump are shut off.

D. If only S1 is activated, fuel is supplied and the pump, if on, shuts off.

E. If S2 and S3 are activated and S1 is not, the result must be the same as if only S1 were activated.

F. If only S3 is activated, fuel must be supplied and the pump must be on.

G. If only S2 is activated, both the fuel and the pump must be off.

H. If no switches are activated, fuel must be supplied and the pump must be off.

Using the SSI design algorithm, design a digital circuit to control the above industrial process.

2. Implement your final NAND/NOR/XOR design on the Trainer. Then switch power ON and verify the proper operation of the circuit. Make a serious attempt to design and implement your own circuit. If you have trouble, refer to the following discussion to check your results.

Discussion

Applying the SSI design algorithm, you should get the following results:

I/O Definition — The circuit has three binary input pressure switches S1, S2, and S3. The three switches must control two outputs, the fuel valve and the electrical pump. Label the fuel valve "f" and the pump "p". Consequently, the inputs and outputs have been defined as follows:

Inputs: Three pressure switches: S1, S2, and S3.
Outputs: A fuel valve (f), and a pump (p).

Logic Definition — Using positive logic, let a pressure switch be a logic 1 when it is activated by a high pressure. Likewise, suppose a logic 1 will activate both the fuel valve and the pump. Thus, the logic is defined as follows:

Pressure Switch Inputs: (S1, S2, and S3)	Logic 0 — Switch (S1, S2, S3) Not Activated Logic 1 — Switch (S1, S2, S3) Activated
Fuel Valve and Pump: (f and p)	Logic 0 — Off Logic 1 — On

Truth Table — Applying the above logic definitions to the industrial problem statement, you get the following truth table:

S1	S2	S3	f	p
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	0	1

Note: If you have defined your logic differently, your truth table will most likely be different from the one above. However, this does not mean that you are wrong. Your resulting circuit should still work according to **your** logic definitions.

The Boolean Expression — From the above truth table, you get the following expressions for f and p:

$$f = \overline{S1}\,\overline{S2}\,\overline{S3} + \overline{S1}\,\overline{S2}\,S3 + \overline{S1}\,S2\,S3 + S1\,\overline{S2}\,\overline{S3} + S1\,\overline{S2}\,S3$$

$$p = \overline{S1}\,\overline{S2}\,S3 + S1\,\overline{S2}\,S3 + S1\,S2\,S3$$

Simplification — Using Karnaugh mapping, the above expressions simplify to:

$$f = \overline{S2} + \overline{S1}\,S3$$

$$p = S1\,S3 + \overline{S2}\,S3 = S3\,(S1 + \overline{S2})$$

AND/OR Design — The AND/OR design required to implement the two expressions is shown in Figure E6-1a. Notice that the circuit has three inputs (S1, S2, and S3) and two outputs (f and p).

NAND/NOR/XOR Design — A complete NAND design is shown in Figure E6-1b. This design can be implemented with two ICs: two 74LS00 NAND ICs, or one 74LS00 NAND IC and a 74LS04 Inverter IC.

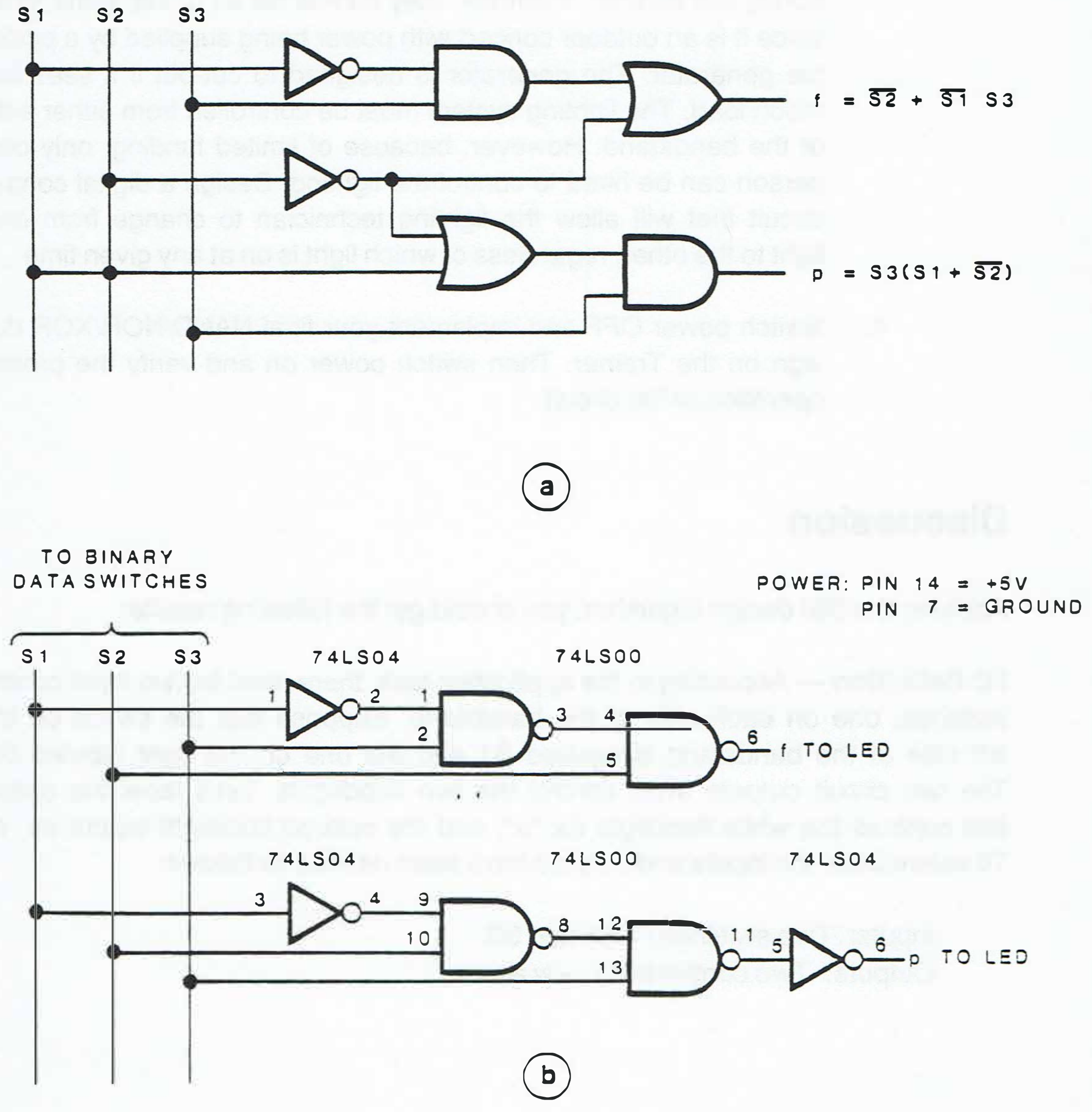


Figure E6-1

The AND/OR design (a) and NAND design (b) for the industrial control problem.

Procedure (continued)

3. Here is another application problem:

   You must design a digital controlled concert lighting system that meets the following criteria:

   The bandstand must be illuminated by two floodlights, one white and one colored. At least one of the lights must be on at all times during the concert. However, they cannot be on at the same time, since it is an outdoor concert with power being supplied by a portable generator. The generator is designed to cut-out if it sees too much load. The lighting system must be controlled from either side of the bandstand. However, because of limited funding, only one person can be hired to control the lighting. Design a digital control circuit that will allow the lighting technician to change from one light to the other, regardless of which light is on at any given time.

4. Switch power OFF and implement your final NAND/NOR/XOR design on the Trainer. Then switch power on and verify the proper operation of the circuit.

Discussion

Applying the SSI design algorithm, you should get the following results:

I/O Definition — According to the application task, there must be two input control switches, one on each side of the bandstand. Suppose that the switch on the left side of the bandstand is labeled S1 and the one on the right labeled S2. The two circuit outputs must control the two floodlights. Let's label the output that controls the white floodlight as "w", and the colored floodlight output as "c". To summarize, the inputs and outputs have been defined as follows:

Inputs: Two switches — S1 and S2.
Outputs: Two control lines — w and c.

Logic Definition — Again, using positive logic, suppose an input switch generates a logic 1 when it is in the up position and a logic 0 when it is in its down position. Also, suppose a logic 1 will turn a floodlight on, while a logic 0 will turn it off. Thus, the logic is defined as follows:

Bandstand Control Switches: (S1 and S2)	Logic 0 — Down Logic 1 — Up
Floodlight Control Lines: (w and c)	Logic 0 — Off Logic 1 — On

Truth Table — From the above logic definitions and the problem statement, you get the following truth table:

S1	S2	w	c
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	0

To get the truth table, you must assume that one of the lights is on and the other off. Then, each time a switch is thrown, the lighting reverses.

Boolean Expression — From the truth table, you get:

$$w = \overline{S1}\,\overline{S2} + S1\,S2$$

$$c = \overline{S1}\,S2 + S1\,\overline{S2}$$

Simplification — Here is where the application of a little common-sense pays off. Look at the above truth table. Do you see any particular logic operations? Doesn't the white light respond to even switch parity and the colored light respond to odd parity? As a result, the white light requires an XNOR operation and the colored light requires an XOR operation, or:

$$w = \overline{S1 \oplus S2}$$

$$c = S1 \oplus S2$$

Final Design — Following simplification, the final design reduces to two gates: a 2-bit XNOR gate and a 2-bit XOR gate as shown in Figure E6-2.

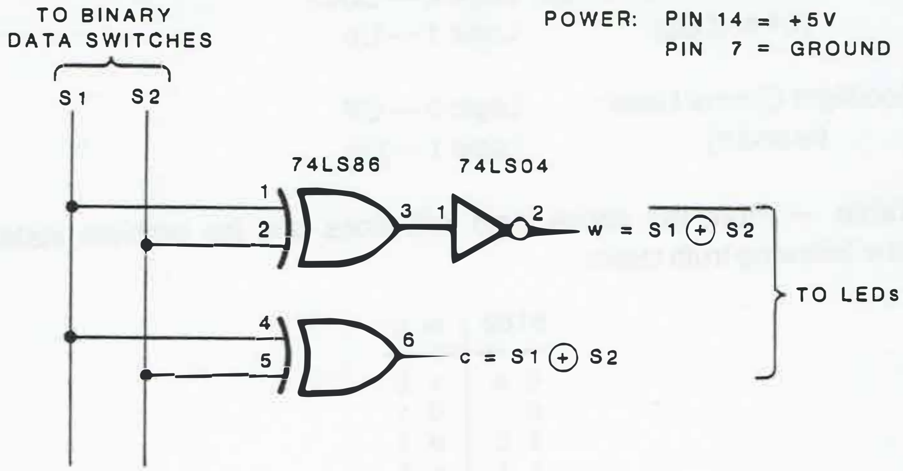


Figure E6-2

Solution to the concert lighting problem.

Procedure (continued)

5. Switch power OFF and remove the circuit components and wire.

6. Proceed to Experiment 7.

EXPERIMENT 7

COMBINATIONAL LOGIC DESIGN USING SSI DEVICES — PART 2

Objectives:

To apply the SSI design algorithm to combinational logic design problems.

To implement your design and verify that it meets the application task.

Introduction

This experiment is an extension of the last experiment where you were given an application problem and applied the SSI design algorithm to create a working digital control circuit. You will find the two problems in this experiment a bit more challenging. However, a little common-sense along with careful application of each design step will lead you to a successful control circuit. If necessary, refer to the Introduction in Experiment 6 for a summary of the design steps.

Materials Required

Heathkit Engineering Design Trainer ET-1000
3 — 74LS00 Quad 2-bit NAND
2 — 74LS02 Quad 2-bit NOR
1 — 74LS04 Hex Inverter

Procedure

1. You must design a digital climate control system for an automobile to meet the following criteria:

   The digital circuit must control a heater and air conditioning unit according to the temperature conditions sensed by a thermostat. The thermostat will either call for heat or air conditioning, not both. In addition, to protect the car battery, a special circuit senses the battery voltage and disables the climate control system if a low battery voltage condition is detected. Thus, neither the heater nor the air conditioner will come on when a low battery condition exists. However, to protect the customer from extreme hot or cold weather conditions, you must provide a dashboard switch that can be used to override the low battery voltage detector and allow the climate control system to operate normally.

   Use the SSI design algorithm to design a digital climate control system.

2. Implement your final NAND/NOR/XOR design on the Trainer. Then switch power ON and verify the proper operation of the circuit. Refer to the following discussion to check your results.

Discussion

Applying the SSI design algorithm, you should get the following results:

I/O Definition — The circuit has three inputs, the thermostat, low battery voltage detector, and a dashboard override switch. Let's label these inputs as T, B, and S, respectively. There must be two output signals generated by the circuit, one to control the heater and one for the air conditioner. Label the heater signal line "h" and the air conditioner signal line "a". Thus, the inputs and outputs have been defined as follows:

Inputs: Thermostat — T
Low Battery Voltage Detector — B
Dashboard Switch — S

Outputs: Heater — h
Air Conditioner — a

Logic Definition — The job of the thermostat is to call for either heat or air conditioning, not both. So, suppose that it generates a logic 1 when heat is required, and a logic 0 for air conditioning. The system must be disabled if a low battery voltage is detected, unless it is overridden by the dashboard switch. Suppose the low battery voltage detector generates a logic 1 if the battery is okay and a logic 0 if the battery is low. To override the low battery detector the dashboard switch will generate a logic 1, otherwise it supplies a logic 0 to the circuit.

The two output signals must turn the heater and air conditioner on and off. In both cases, let a logic 1 output turn the respective unit on and a logic 0 output turn it off. To summarize, the I/O logic has been defined as follows:

Thermostat Input (T):	Logic 0 = Air Conditioning Logic 1 = Heat
Low Battery Input (B):	Logic 0 = Battery Low Logic 1 = Battery Okay
Dashboard Override Switch (S):	Logic 0 = No Override Logic 1 = Override

Truth Table — From the above logic definitions and the problem statement, you get the following truth table:

T	B	S	a	h
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

Boolean Expression — Writing the sum-of-products expression from the above truth table you get:

$$a = \overline{T}\,\overline{B}S + \overline{T}B\overline{S} + \overline{T}BS$$

$$h = T\overline{B}S + TB\overline{S} + TBS$$

Simplification — Applying Boolean algebra or Karnaugh mapping to simplify the above expressions results in:

$$\begin{aligned} h &= TB + TS \\ &= T(B + S) \end{aligned}$$

$$\begin{aligned} a &= \bar{T}B + \bar{T}S \\ &= \bar{T}(B + S) \end{aligned}$$

AND/OR Design — The AND/OR designs required to implement the above expressions are shown in Figure E7-1a. The circuit has three inputs (T, B, and S), and two outputs (h and a).

NAND/NOR/XOR Design — A complete NAND design is shown in Figure E7-1b. This design can be implemented using two ICs: a 74LS00 NAND IC and a 74LS04 Hex Inverter IC.

Procedure (continued)

3. Use the SSI design algorithm to design a four input voter circuit.

   The circuit outputs must indicate if an issue passes, fails, or ties.

4. Switch power OFF and implement your final design on the Trainer. Then switch power ON and verify the proper operation of the circuit.

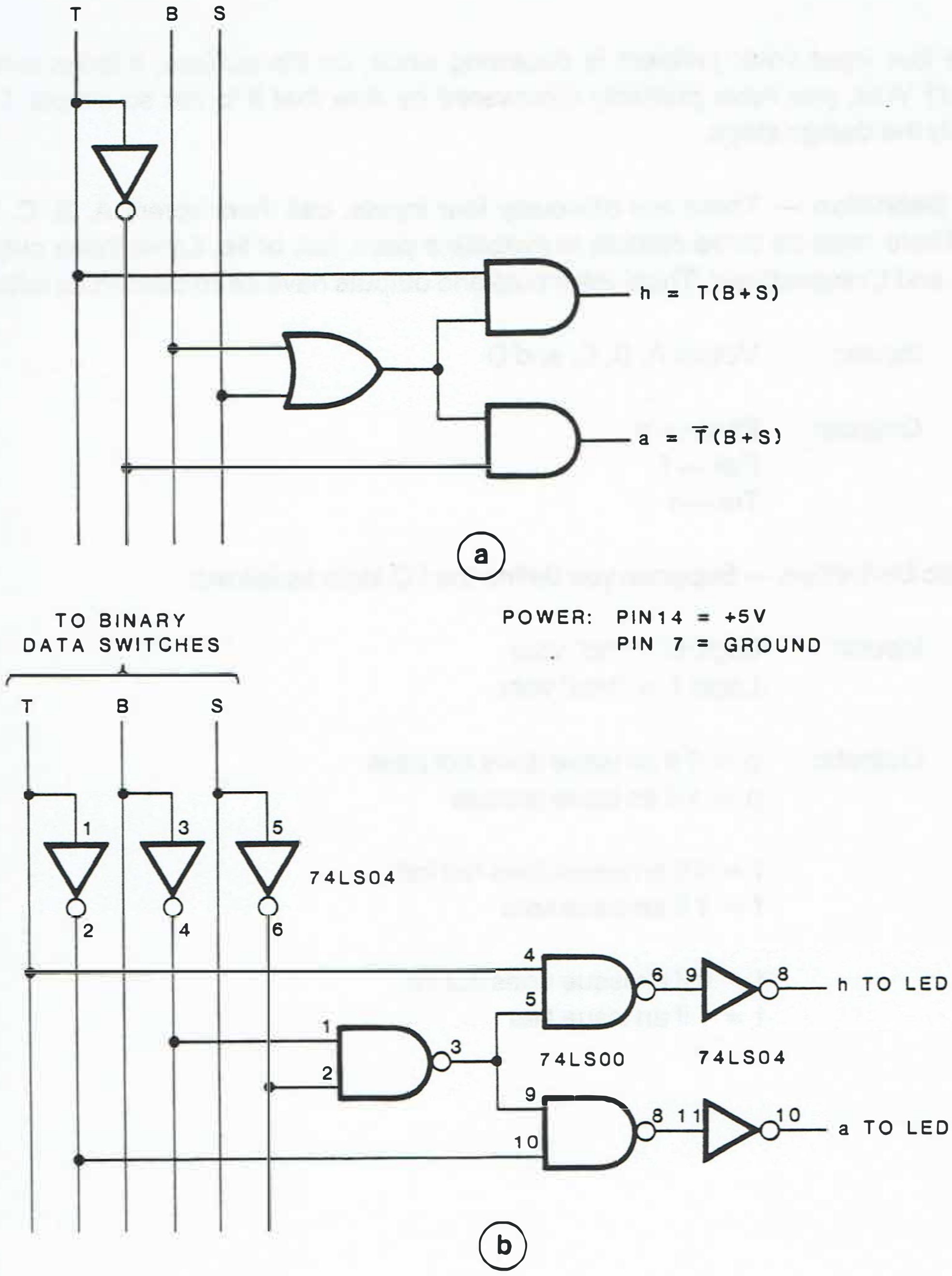


Figure E7-1

The AND/OR design (a) and NAND design (b) solutions for the climate control problem.

Discussion

The four input voter problem is deceiving since, on the surface, it looks simple, right? Well, you have probably discovered by now that it is not so simple. Let's apply the design steps.

I/O Definition — There are obviously four inputs, call them voters A, B, C, and D. There must be three outputs to indicate a pass, fail, or tie. Label these outputs p, f, and t, respectively. Thus, the inputs and outputs have been defined as follows:

Inputs: Voters A, B, C, and D

Outputs: Pass — p
Fail — f
Tie — t

Logic Definition — Suppose you define the I/O logic as follows:

Inputs: Logic 0 = "no" vote
Logic 1 = "yes" vote

Outputs: p = 0 if an issue does not pass
p = 1 if an issue passes

f = 0 if an issue does not fail
f = 1 if an issue fails

t = 0 if an issue does not tie
t = 1 if an issue ties

Truth Table — Using the above logic definitions, you get:

A	B	C	D	p	f	t
0	0	0	0	0	1	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	0	0	1
0	1	1	0	0	0	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	0	1
1	0	1	0	0	0	1
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	0	0

Boolean Expression — At this point, you could plunge right in and write the Boolean expressions from the above truth table, simplify them using K-maps, and implement them using a NAND/NOR/XOR design. If you do, you end-up with a design that is quite complex. Good engineering and good common-sense go together. Here is where a little common-sense really pays off in the design concept. How does a tie relate to a pass and a fail? Isn't a tie NOT a pass AND NOT a fail? In other words, when an issue does NOT pass AND does NOT fail you get a tie, right? So, using this idea, you can immediately write a very simple Boolean expression for the tie output as follows:

$$t = \overline{p}\,\overline{f}$$

This says that you can generate the tie output by inverting the pass and fail outputs and ANDing them together.

The pass and fail output expressions are derived from the truth table as follows:

$$p = \overline{A}BCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D} + ABCD$$
$$f = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}C\overline{D} + \overline{A}B\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}\,\overline{D}$$

Simplification — Demorgan's first theorem can be applied to the tie (t) expression to get:

$$t = \overline{p+f}$$

Using Karnaugh mapping, you get the following simplified expressions for the pass (p) and fail (f) outputs:

$$\begin{aligned} p &= ABC + ABD + ACD + BCD \\ &= AB(C + D) + CD(A + B) \end{aligned}$$

$$\begin{aligned} f &= \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{D} + \overline{A}\,\overline{C}\,\overline{D} + \overline{B}\,\overline{C}\,\overline{D} \\ &= \overline{A}\,\overline{B}(\overline{C} + \overline{D}) + \overline{C}\,\overline{D}(\overline{A} + \overline{B}) \end{aligned}$$

AND/OR Design — The AND/OR designs required to implement the above expressions are shown in Figure E7-2. Notice that there are four inputs (A, B, C, and D) representing the four voters. There are three outputs to indicate a pass (p), fail (f), or tie (t). The tie output is obtained by NORing the pass and fail outputs.

NAND/NOR/XOR Design — The final design shown in Figure E7-3 can be implemented with six ICs: three 74LS00 NAND ICs, two 74LS02 NOR ICs, and a 74LS04 Inverter IC.

Procedure (continued)

5. Switch power OFF and remove the circuit components and wire.
6. Return to Unit 2 and complete the Unit Examination.

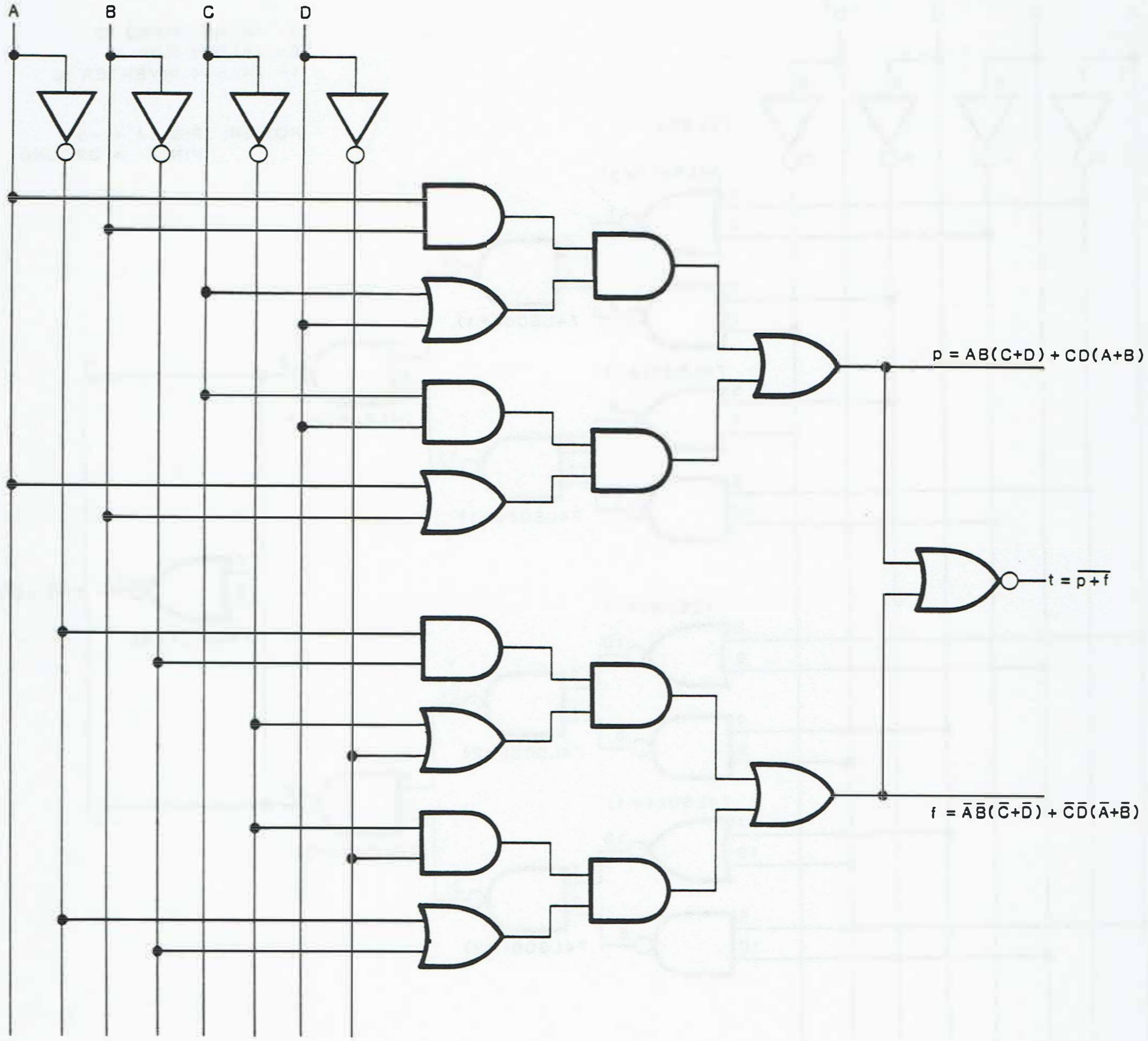

A
B
C
D
$p = AB(C+D) + CD(A+B)$
$t = \overline{p+f}$
$f = \bar{A}\bar{B}(\bar{C}+\bar{D}) + \bar{C}\bar{D}(\bar{A}+\bar{B})$

Figure E7-2

The AND/OR design for the 4-input voter problem.

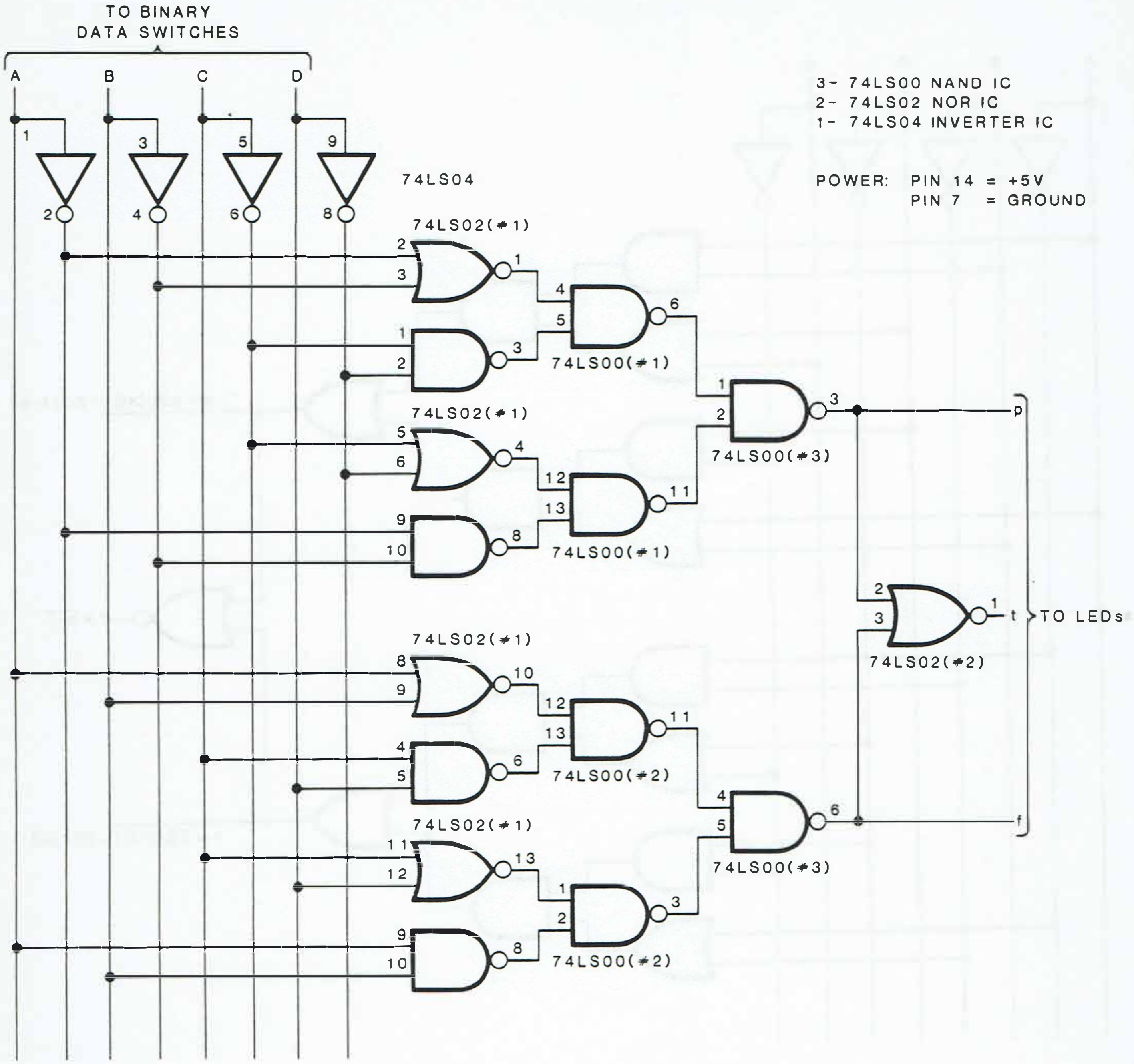


Figure E7-3

The NAND/NOR solution to the 4-input voter problem.

EXPERIMENT 8

MSI DEVICES

Objectives:

To demonstrate the operation of an MSI multiplexer.

To demonstrate the operation of an MSI decoder.

To demonstrate the operation of an MSI demultiplexer.

Introduction

Multiplexers and demultiplexers can be thought of as electronic switches. A multiplexer selects one of many inputs and routes it to a single output. Conversely, a demultiplexer takes a single input and distributes it to one of many outputs. MSI demultiplexers can also be made to operate as decoders. A decoder detects a specific binary input value by activating a unique output line for a given binary input.

Multiplexers, demultiplexers, and decoders are very common MSI devices. Multiplexers and decoders are particularly useful for implementing combinational logic. In addition, MSI decoders are used in computer systems for address decoding. In this experiment you will be acquainted with the operation of each of these MSI devices. This will prepare you for the next experiment, where you use them to implement combinational logic.

Materials Required

Heathkit Engineering Design Trainer ET-1000
1 — 74LS153 Dual 4:1 Multiplexer
1 — 74LS138 1-of-8 Decoder

Procedure

1. Construct the 74LS153 multiplexer circuit shown in Figure E8-1. Connect the multiplexer to the Trainer as follows:

   A. Set the PULSE (SEC) switch in the Trainer Function Generator section to 2m and connect wires from this output to LED L3 and the D3 input (pin 3) of the multiplexer.

   B. Set both the FREQ. control and FREQ. MULTIPLIER switch in the Trainer Function Generator section to 1.

   C. Connect wires from the left (positive/negative transition) square wave output in the Trainer Function Generator to LED L2 and the D2 input (pin 4) of the multiplexer.

   D. Connect wires from the right (negative/positive transition) square wave output in the Trainer Function Generator to LED L1 and the D1 input (pin 5) of the multiplexer.

   E. Set Binary Data switch 0 to a logic 1 and connect a wire from this switch to the D0 input (pin 6) of the multiplexer.

   F. Connect wires from the A and B spring-loaded logic switches on the Trainer to the A and B address inputs (pins 2 and 14), respectively, of the multiplexer.

   G. Connect the y1 output of the multiplexer (pin 7) to LED L7.

2. Switch power ON an observe the LEDs. The address input to the multiplexer is AB = 00, since the logic switches are not activated. You should see a constant logic 1 output from the multiplexer at LED L7. Why? ____________________
____

3. Generate an address input of AB = 01 to the multiplexer by holding the B logic switch to the left. You should see LED L7 pulse on and off at a 1 second rate. Why? ____________________
____

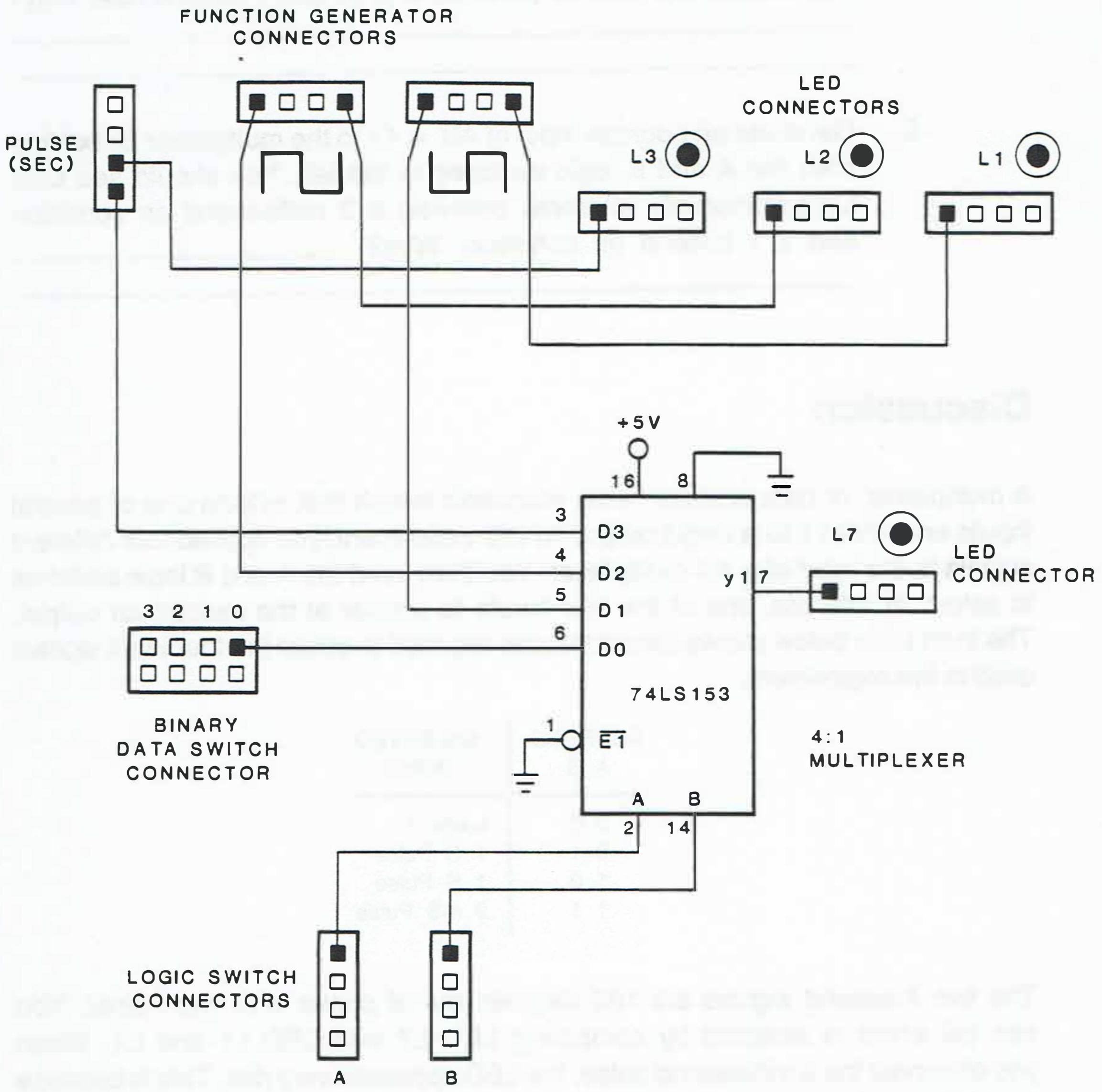


Figure E8-1

A 4:1 multiplexer.

4. Release the B logic switch and generate an address input of AB = 10 to the multiplexer by holding the A logic switch to the left. You should see LED L7 pulse on and off at a 1 second rate. Why? ____________________
____

5. Generate an address input of AB = 11 to the multiplexer by holding both the A and B logic switches to the left. You should see LED L7 continuously alternate between a 2 millisecond on condition and a 1 second off condition. Why? ____________________
____

Discussion

A multiplexer, or data selector, is an electronic switch that selects one of several inputs and routes it to a single output. In this experiment, you applied four different signals to the input of a 4:1 multiplexer. You then used the A and B logic switches to select, or address, one of the four inputs to appear at the multiplexer output. The truth table below shows the addresses required to select the four input signals used in this experiment.

ADDRESS A B	SELECTED INPUT
0 0	Logic 1
0 1	1 S Pulse
1 0	1 S Pulse
1 1	2 mS Pulse

The two 1-second signals are 180 degrees out of phase with each other. You can tell which is selected by comparing LED L7 with LED L1 and L2. When you observed the 2 millisecond pulse, the LED appeared very dim. This is because the LED is on for such a short period of time.

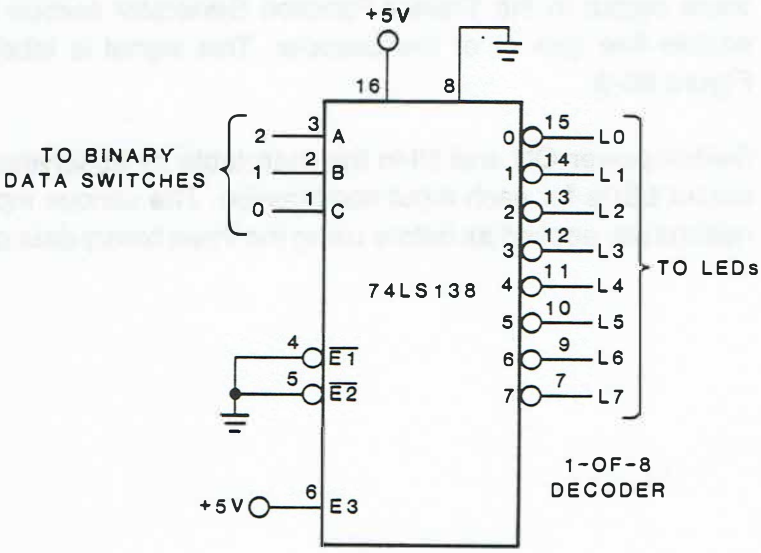
Procedure (continued)

6. Switch power OFF and remove the multiplexer circuit. Then construct the 74LS138 decoder circuit shown in Figure E8-2. Connect the decoder to the Trainer as follows:

    A. Connect the A, B, and C decoder inputs to the 2, 1, and 0 binary data switches, respectively.

    B. Connect decoder output lines 0 through 7 to the L0 through L7 LEDs on the trainer, respectively.

7. Switch power ON. Then fill-in the truth table in Figure E8-2 by observing the eight output LEDs for each input combination. You must generate the various input logic combinations using the three binary data switches.



A	B	C	0	1	2	3	4	5	6	7
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								

Figure E8-2

A 1-of-8 decoder.

Discussion

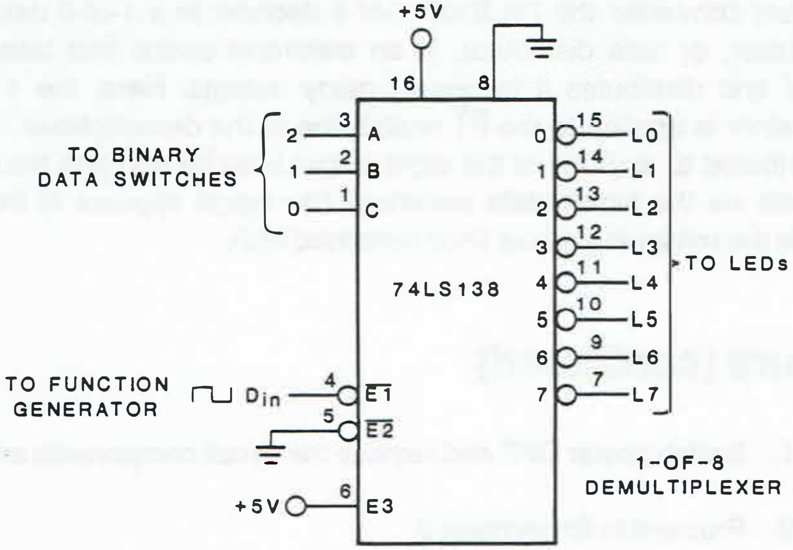
A decoder detects the presence of a specific binary input by activating a unique output line for a given input value. The 74LS138 decoder used in this experiment is a 1-of-8 decoder that detects one of eight input values by generating a logic 0 on the corresponding output line. The truth table that you generated during the experiment demonstrates this operation. Notice that the three enable lines ($\overline{E1}$, $\overline{E2}$, and E3) must be hardwired to an active state for the decoder to operate. You might try removing one of these lines to see what happens.

Procedure (continued)

8. Switch power OFF and disconnect the wire between ground and the $\overline{E1}$ enable line (pin 4) of the decoder.

9. Connect a wire from the left (positive/negative transition) square wave output in the Trainer Function Generator section to the $\overline{E1}$ enable line (pin 4) of the decoder. This signal is labeled D_{in} in Figure E8-3.

10. Switch power ON and fill-in the truth table by observing the eight output LEDs for each input combination. The various input combinations are applied as before using the three binary data switches.



A	B	C	0	1	2	3	4	5	6	7
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								

Figure E8-3

A 1-of-8 demultiplexer.

Discussion

You have just converted the 74LS138 1-of-8 decoder to a 1-of-8 demultiplexer. A demultiplexer, or data distributor, is an electronic switch that takes a single input signal and distributes it to one of many outputs. Here, the 1 Hz signal from the Trainer is applied to the $\overline{E1}$ enable line of the demultiplexer. This signal is then distributed to any one of the eight output lines by applying the respective address code via the binary data switches. The signal appears at the selected output, while the remaining output lines remained high.

Procedure (continued)

11. Switch power OFF and remove the circuit components and wire.
12. Proceed to Experiment 9.

EXPERIMENT 9

COMBINATIONAL LOGIC DESIGN USING MSI DEVICES

Objectives:

To design digital control circuits using MSI multiplexers.

To design digital control circuits using MSI decoders.

To compare SSI and MSI designs for the same control circuit.

To demonstrate that MSI designs are more economical to implement when the design has four or more inputs.

Introduction

In Unit 3, you learned how to implement combinational logic using MSI multiplexers and decoders. In this experiment you will apply this knowledge by designing MSI circuits to implement the automobile climate control system and the four input voter circuit from Experiment 7. You will find that the climate control system is implemented more economically with SSI logic, while the voter circuit is implemented more economically using MSI logic. Can you guess why this might be the case?

Make a sincere effort to design and implement **your own** circuits by referring to the MSI design material in Unit 3. If you have trouble, refer to the experiment discussions. These have been written to lead you through the design process.

Materials Required

Heathkit Engineering Design Trainer ET-1000
1 — 74LS00 Quad 2-bit NAND
1 — 74LS02 Quad 2-bit NOR
1 — 74LS04 Hex Inverter
1 — 74LS10 Triple NAND
1 — 74LS138 1-of-8 Decoder
1 — 74LS153 Dual 4:1 Multiplexer

Procedure

1. In Experiment 7, you designed an SSI circuit to implement the automobile climate control system. Now, design an MSI **multiplexer** circuit to implement the same control system using the n – 1 design procedure.

2. Implement your design using a 74LS153 Dual 4:1 multiplexer and the required SSI logic. Then switch power ON and verify the proper operation of the circuit.

   How does this design compare to your SSI design in Experiment 7?

   ____

Discussion

Recall that the n – 1 multiplexer design solution requires you to break-up the truth table into pairs of input variables. The truth table for the automobile climate control system is:

T	B	S	a	h
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

Ignoring the most significant input, T, there are four pairs of the B and S inputs as follows:

	INPUT	B S	a h
Pair 0	0	0 0	0 0
	4	0 0	0 0
Pair 1	1	0 1	1 0
	5	0 1	0 1
Pair 2	2	1 0	1 0
	6	1 0	0 1
Pair 3	3	1 1	1 0
	7	1 1	0 1

The next step is to assign a pair value for each pair. Recall that, for a given output, the pair values are assigned according to how the output changes within the pair as follows:

For a given output, y, and unused input, A:

y = 00, pair value = 0
y = 01, pair value = A
y = 10, pair value = $\overline{A}$
y = 11, pair value = 1

Translating this for the h and a outputs and the unused T input, the pair values are:

	a OUTPUT	h OUTPUT
Pair 0	0	0
Pair 1	$\overline{T}$	T
Pair 2	$\overline{T}$	T
Pair 3	$\overline{T}$	T

Since there are two outputs and four pairs, two 4:1 multiplexers are required to implement the design. The 74LS153 is ideal for this application, since it is a dual 4:1 multiplexer. The pair values must be applied to the respective input lines of each multiplexer section as shown in Figure E9-1. The remaining inputs, B and C, are connected to the multiplexer address lines as shown.

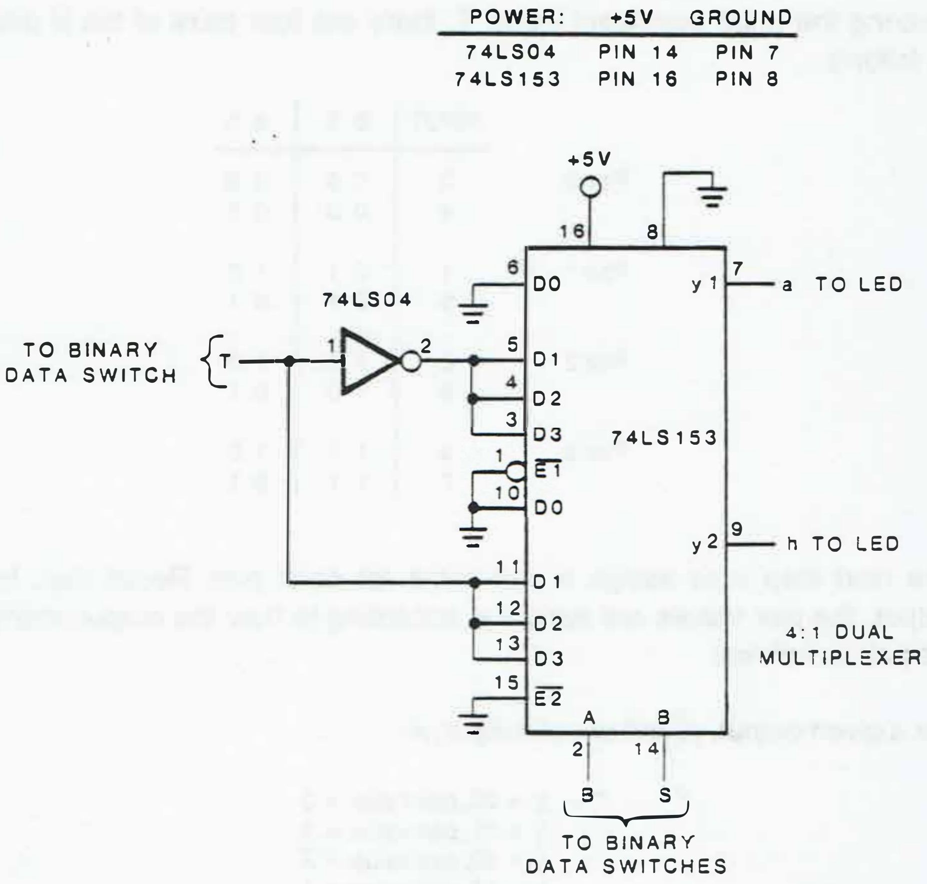


Figure E9-1

A multiplexer implementation of the climate control system.

The difference between the MSI multiplexer design and the SSI design is that the MSI design requires a 75LS153 multiplexer while the SSI design requires a 74LS00 NAND. Both require a 74LS04 inverter IC. So, the chip count is the same. How about the cost? The NAND IC is about half the cost of the multiplexer, thereby making the SSI design more economical to implement.

Procedure (continued)

3. Design an MSI **decoder** circuit to implement the automobile climate control system.
4. Switch power OFF and implement your design using a 74LS138 1-of-8 decoder and the required SSI logic. Then switch power ON and verify the proper operation of the circuit.

   How does this design compare to the above multiplexer design and the SSI design in Experiment 7? ____________________

   ____

Discussion

The decoder design process requires that you first write a numerical expression for each circuit output. Using the preceding heater and air conditioning truth table, the correct expressions are:

$$a = \Sigma(1,2,3)$$
$$h = \Sigma(5,6,7)$$

Next, a 1-of-8 decoder must be used, since there are three inputs (if necessary, refer to Table 3-3). The three circuit inputs, T, B, and S are connected directly to the A, B, and C inputs of the decoder as shown in Figure E9-2. Decoder output lines 1, 2, and 3 must drive a 3-bit NAND gate to get the air conditioning (a) output, while lines 5, 6, and 7 drive another 3-bit NAND gate to get the heater (h) output.

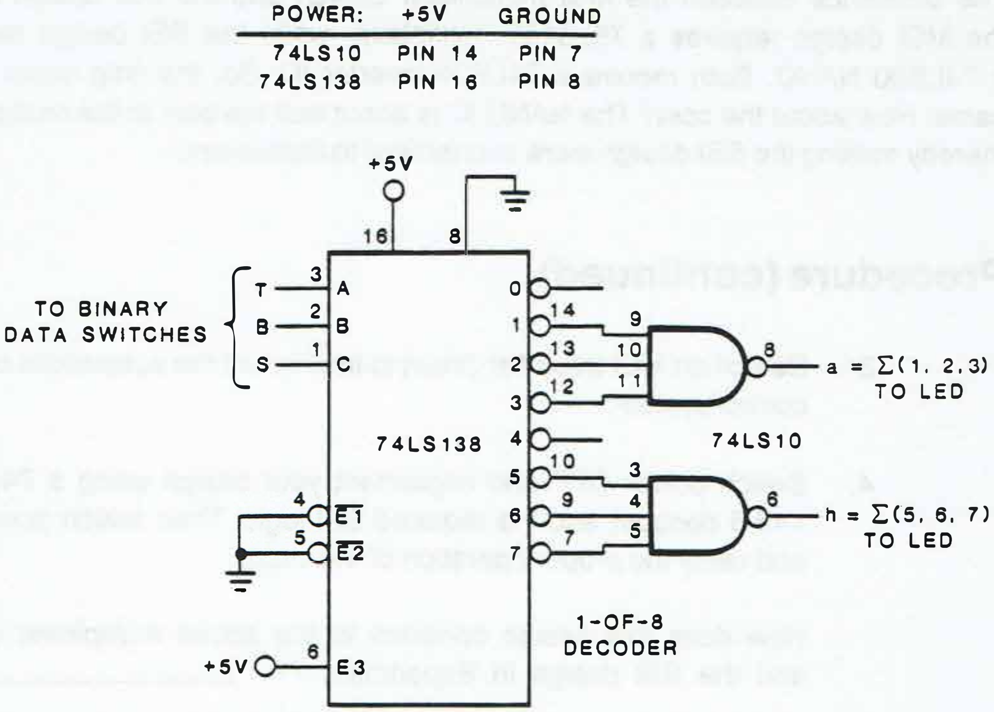


Figure E9-2

A decoder implementation of the climate control system.

The decoder design requires two ICs: a 74LS138 decoder and a 75LS10 3-input NAND IC. This results in the same chip-count as the previous two designs. However, the cost of the 74LS138 decoder is about one and one-half times the cost of the 74LS153 multiplexer. Thus, the most economical design to implement is still the SSI design from Experiment 7. You are probably wondering: When does an MSI design become more economical than an SSI design? The next part of this experiment will answer that question.

Procedure (continued)

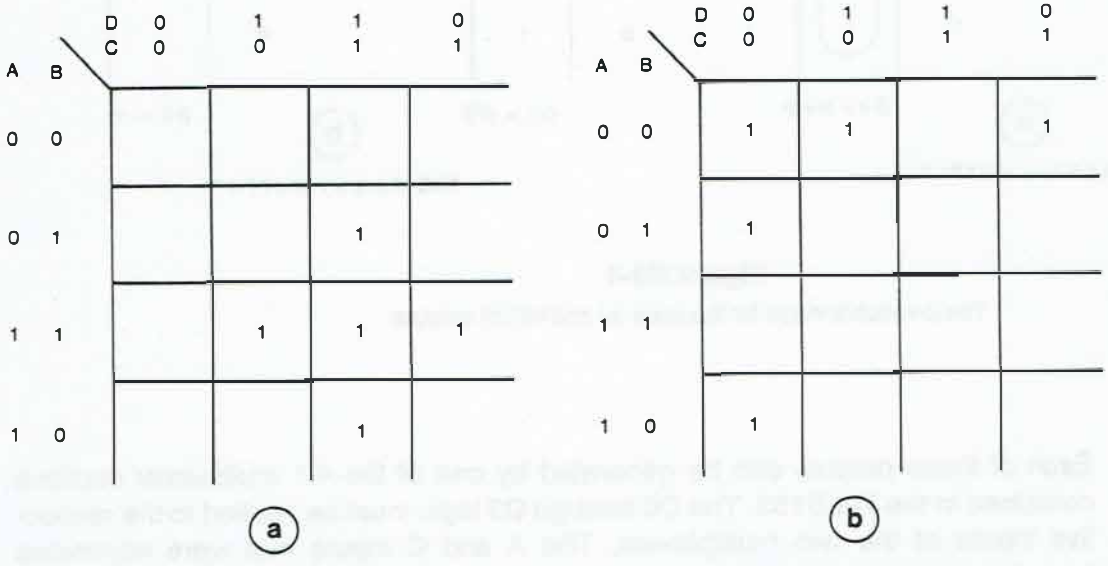
5. Design a multiplexer circuit to implement the four input voter circuit using the n – 2 multiplexer design procedure.

6. Implement your design with a 74LS153 multiplexer and the required SSI logic. Verify the proper operation of the circuit.

   How does this design compare to the SSI design in Experiment 7? ______________________________

Discussion

The n – 2 design procedure first requires you to construct a K-map for each output. The K-maps obtained for the pass (p) and fail (f) outputs of the four input voter problem are shown in Figure E9-3. Remember, the tie (t) output is obtained by NORing the p and f outputs. The next step is to break-up each K-map into four two-variable sections. The possible sections are shown in Unit 3, Figure 3-14. I have chosen to use the grid in Figure 3-14b that eliminates the A and C inputs. Actually, any of the grids could be used, since they all result in the same number of groupings.



AB \ DC	00	10	11	01
00				
01			1	
11		1	1	1
10			1	

ⓐ THE PASS (p) K-MAP

AB \ DC	00	10	11	01
00	1	1		1
01	1			
11				
10	1			

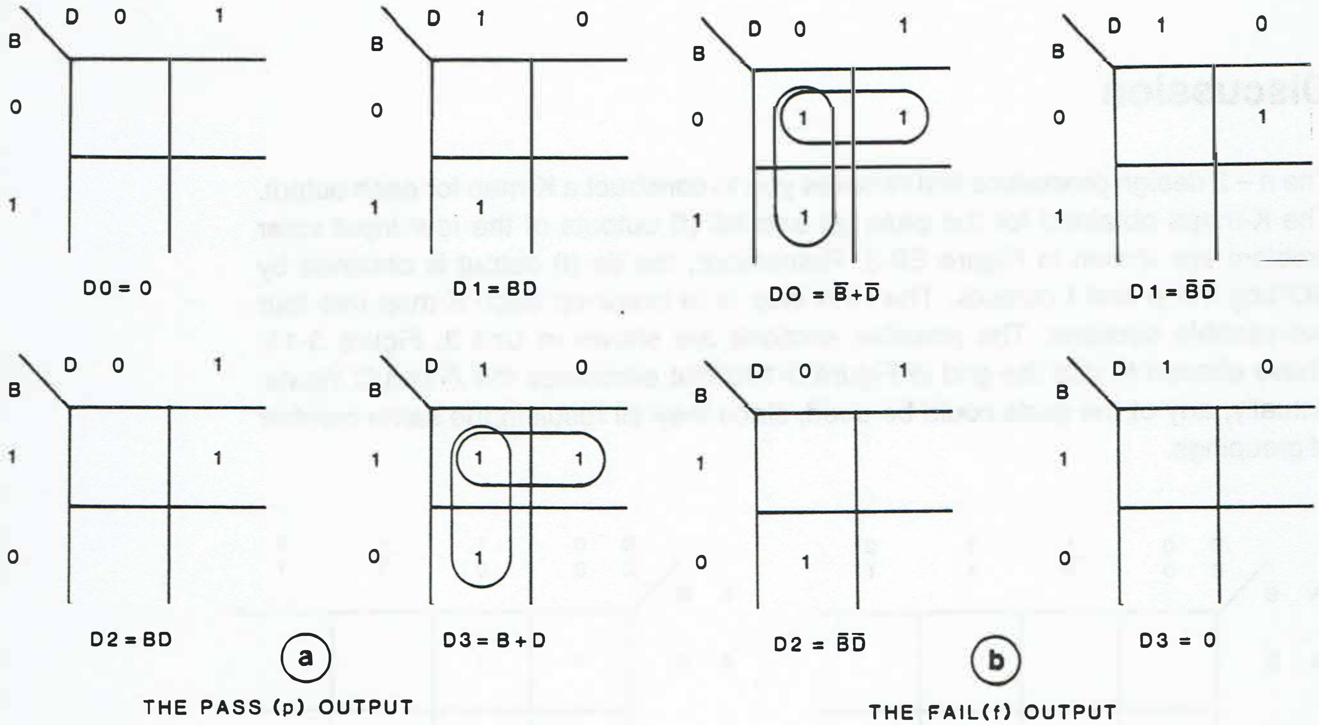
ⓑ THE FAIL (f) K-MAP

Figure E9-3

The K-maps for the pass (a) and fail (b) outputs of the 4- input voter circuit.

The two-variable sections of Figure 3-14b are shown in Figure E9-4 with the corresponding plots transferred from both the "p" and "f" K-maps. Grouping the two-variable plots and reading the groups you get:

	p OUTPUT	f OUTPUT
D0	0	$\overline{B}+\overline{D}$
D1	BD	$\overline{B}\,\overline{D}$
D2	BD	$\overline{B}\,\overline{D}$
D3	B+D	0



D0 = 0
D1 = BD
D0 = $\overline{B}+\overline{D}$
D1 = $\overline{B}\,\overline{D}$
D2 = BD
D3 = B+D
D2 = $\overline{B}\,\overline{D}$
D3 = 0
a
b
THE PASS (p) OUTPUT
THE FAIL(f) OUTPUT

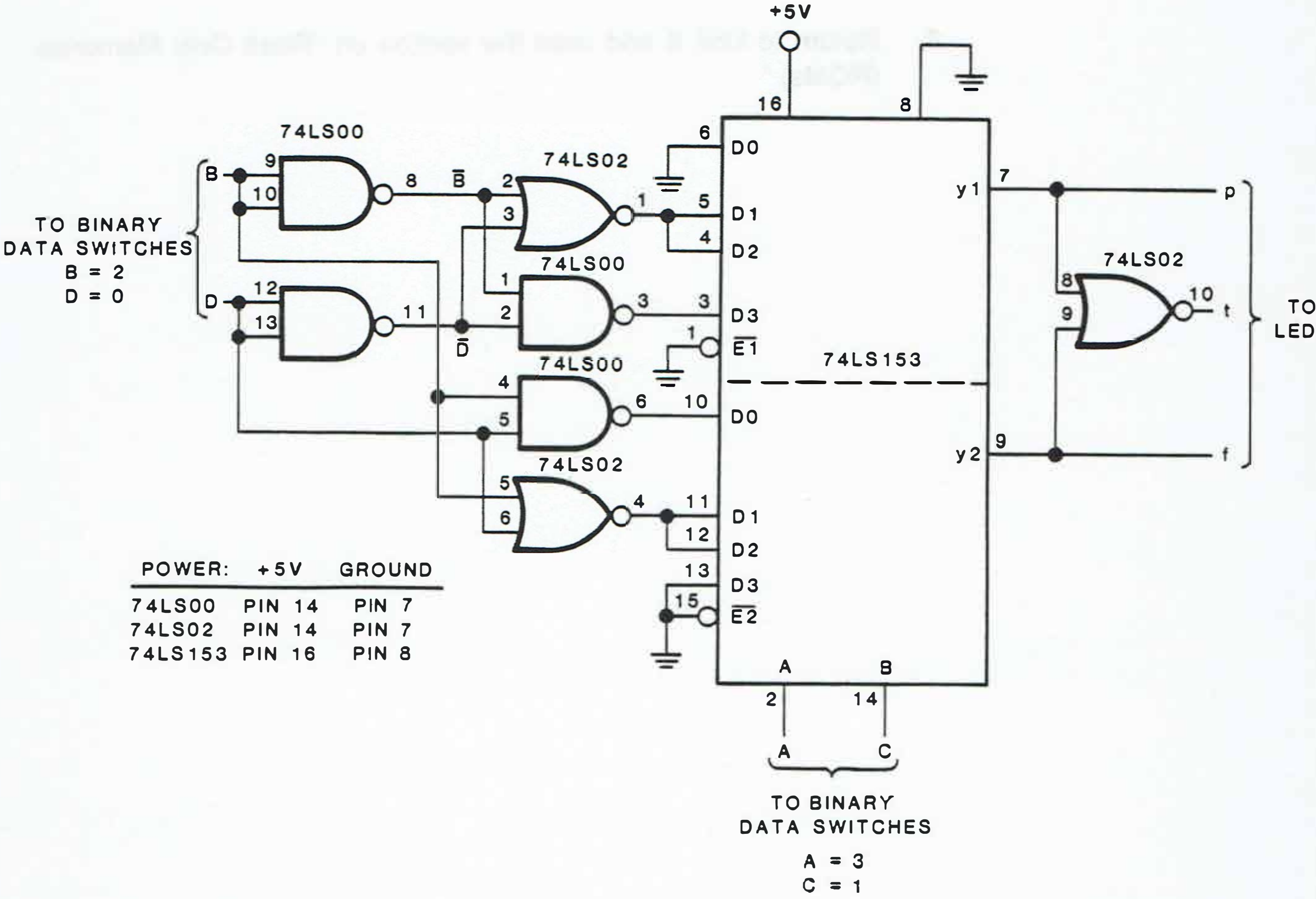
Figure E9-4

The 2-variable maps for the pass (a) and fail (b) outputs.

Each of these outputs can be generated by one of the 4:1 multiplexer sections contained in the 74LS153. The D0 through D3 logic must be applied to the respective inputs of the two multiplexers. The A and C inputs that were eliminated from the K-maps must be tied to the A and B address inputs of the multiplexer IC, respectively.

The resulting design is shown in Figure E9-5. Here, the AND/OR input logic has been converted to NAND/NOR logic. In addition, you will note that the pass and fail outputs have been NORed to generate the tie output.

The n – 2 multiplexer design in Figure E9-5 requires three ICs: a 74LS153, 74LS00, and 74LS02. Notice that the two inverters are constructed from the two extra NAND gates in the 74LS00. The SSI design in Experiment 7 required five ICs. Thus, the MSI design has decreased the chip count by two. In addition, the cost to implement the MSI design is about 25% less than the SSI design.



POWER:	+5V	GROUND
74LS00	PIN 14	PIN 7
74LS02	PIN 14	PIN 7
74LS153	PIN 16	PIN 8

Figure E9-5

The n – 2 multiplexer design solution for the 4-input voter problem.

Here is where an MSI design pays-off. Usually, you will find that MSI designs are not more economical to implement than SSI designs unless you have at least four inputs. This has been demonstrated in the experiment. You found it more economical to implement the three input automobile climate control design using SSI logic, but the four input voter circuit is more economical when implemented with MSI logic.

Procedure (continued)

7. Switch power OFF and remove the circuit components and wire.
8. Return to Unit 3 and read the section on "Read Only Memories (ROMs)."

EXPERIMENT 10

COMBINATIONAL LOGIC DESIGN USING A ROM

Objectives:

To demonstrate the operation of a diode ROM.

To show how address decoding reduces the number of input lines required for a ROM.

To implement arithmetic functions using diode ROMs.

Introduction

The simplest way to implement combinational logic is to use a read only memory, or ROM. The ROM simply stores the logic truth table. Consequently, you can go directly from the truth table to ROM implementation and avoid logic equations, simplification, and all the design tricks associated with SSI and MSI designs. The problem with ROM implementation is that it is only cost effective when there are at least four or more input and output variables, and no standard MSI devices are available to implement the design.

In this experiment, you will construct a diode matrix ROM circuit to simulate the action of a ROM IC. The circuit will employ a 1-of-8 decoder to limit the number of address input lines. The ROM circuits that you build will demonstrate how ROMs are used in calculators to implement arithmetic functions.

Materials Required

Heathkit Engineering Design Trainer ET-1000
1 — 74LS04 Hex Inverter
1 — 74LS138 1-of-8 Decoder
8 — 1N4149 Diode

Procedure

1. Construct the diode ROM circuit shown in Figure E10-1 as follows:

   A. Connect the A, B, and C, decoder inputs to binary data switches 2, 1, and 0, respectively. These three switches will be the three ROM address lines A2, A1, and A0, respectively.

   B. Connect decoder output lines 1 through 5 to the diode matrix via 74LS04 inverters as shown. Decoder output lines 0, 6, and 7 are not used.

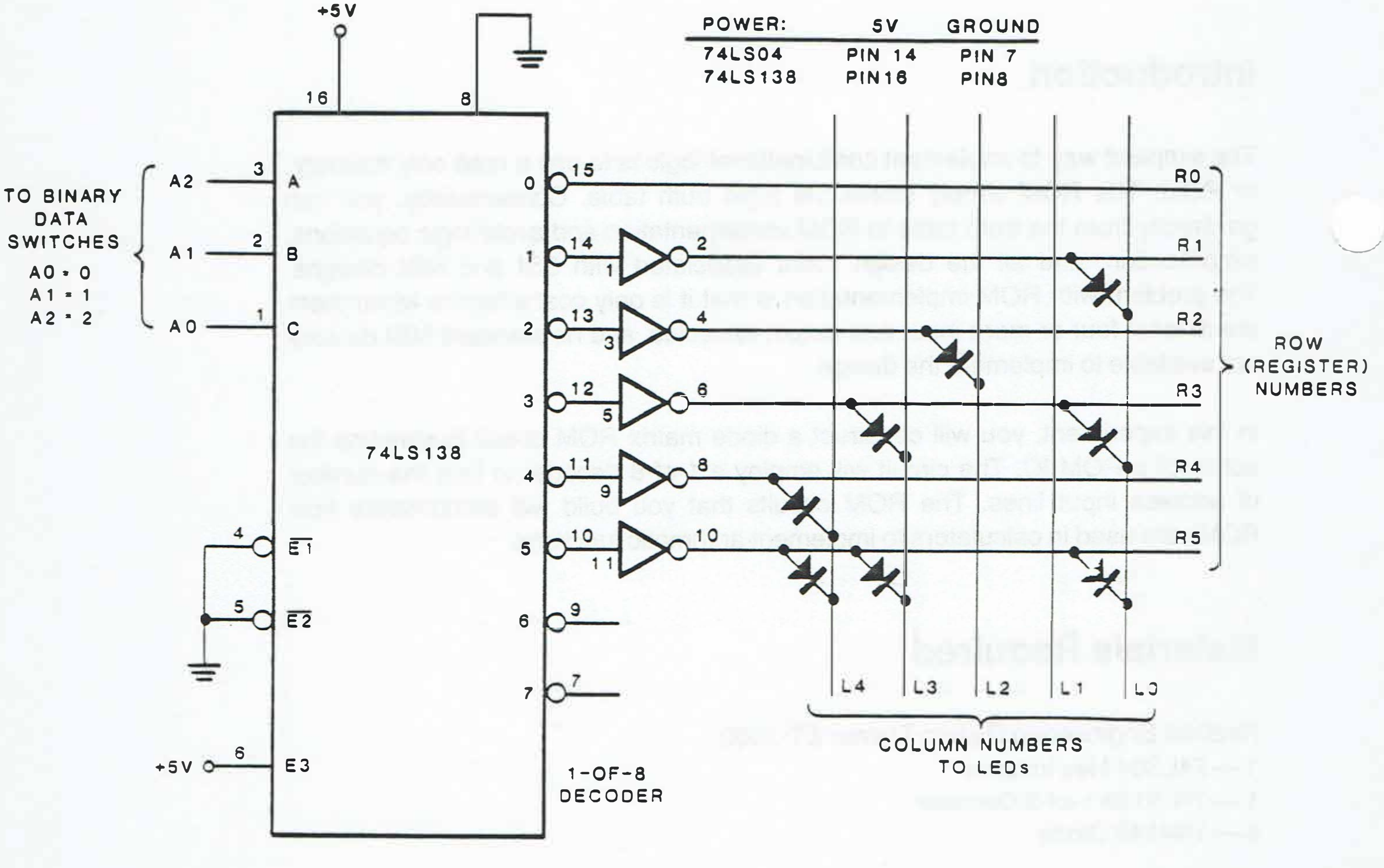


Figure E10-1

A diode ROM.

C. Connect the diodes between the inverter outputs and the various LED indicators on the Trainer as shown. The inverter outputs are connected to the diode anodes and the diode cathodes are connected to the LEDs.

For instance, the output of the bottom inverter must be connected to the anode of three separate diodes. Then, the cathode of one diode is connected to L4, the second to L3, and the third to L0. The cathode end of a diode is marked with a black ring.

2. Switch power ON and fill-in the following truth table by applying the input logic via the three data switches and observing the LED outputs.

A2	A1	A0	L4	L3	L2	L1	L0
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Can you think of the arithmetic function that is being implemented by the circuit? (Hint: Convert the binary input and output values to decimal values. Then, try to determine what arithmetic operation must be performed on the input value to get the corresponding output value.)

Why do you get all 0 outputs for any input value above 101?

Discussion

The diode ROM in Figure E10-1 is a 6 × 5 ROM, since there are six registers, each containing five data bits. A given register is selected and its output appears on the LEDs by applying the appropriate register address to the decoder input lines. For example, an address input of 011 selects the R3 register. Here's how: The decoder generates a logic 0 on output line 3 (row R3) for an address input of 011. The decoder output is inverted to a logic 1 and applied to the anodes of the two diodes in the R3 register. This causes the diodes to conduct and illuminate the L3 and L0 LEDs. The resulting output is 01001, the contents of the register.

There are no diodes in the R0 register. Therefore, an input value of 000 will produce an output of 00000 (no LEDs illuminated).

Notice what the 74LS138 decoder does for you. Without it, you would need six separate input lines, one to select each of the six registers. All commercial ROMs employ **on-chip decoding** to limit the number of address pins required by the chip.

Have you figured out what operation has been implemented by the ROM? How about $f(x) = x^2$, where $0 \leqslant x \leqslant 5$? The output LEDs should indicate the square of the binary input value. For example, an input of 011, or 3, will result in an output of 01001, or 9. However, the ROM will only square input values 0 through 5, since it only has six registers.

Procedure (continued)

3. Design a diode matrix ROM to generate the function f(x) = 3x + 2 for $0 \leqslant x \leqslant 3$. (Hint: You must use decoder output lines 0 through 3 and four LED indicators. Thus, you must construct a 4 × 4 diode matrix.)

4. As always, switch power OFF to construct the circuit. Then switch power ON and verify the proper operation of the circuit by applying input values of 000 through 011 to the decoder. If the diode matrix is wired properly, the LED output should indicate the values 0010, 0101, 1000, and 1011, respectively, for the above input values.

   What happens for any inputs above 011? ______________

   ____

Discussion

Since a ROM implements a truth table directly, the place to begin to design a ROM matrix is with the truth table. If you let the value of x range from 0 to 3 in the function f(x) = 3x + 2, you get:

$$\mathbf{f(x) = 3x + 2}$$
$$f(0) = 3(0) + 2 = 2$$
$$f(1) = 3(1) + 2 = 5$$
$$f(2) = 3(2) + 2 = 8$$
$$f(3) = 3(3) + 2 = 11$$

Translating the above results into a binary truth table gives you:

INPUT (A2 A1 A0)			OUTPUT (L3 L2 L1 L0)			
0	0	0	0	0	1	0
0	0	1	0	1	0	1
0	1	0	1	0	0	0
0	1	1	1	0	1	1

To implement the truth table, you must place a diode at each logic one position in a given output register. For instance, register R0 must have a diode in the bit 1 position, register R1 must have a diode in the bit 0 and bit 2 positions, and so on. The resulting design is shown in Figure E10-2. Notice that the figure only shows the decoder-to-buffer-to-diode-to-LED interface portion of the circuit. The rest of the decoder pins are wired as before.

POWER:	+5V	GROUND
74LS04	PIN 14	PIN 7
74LS138	PIN 16	PIN 8

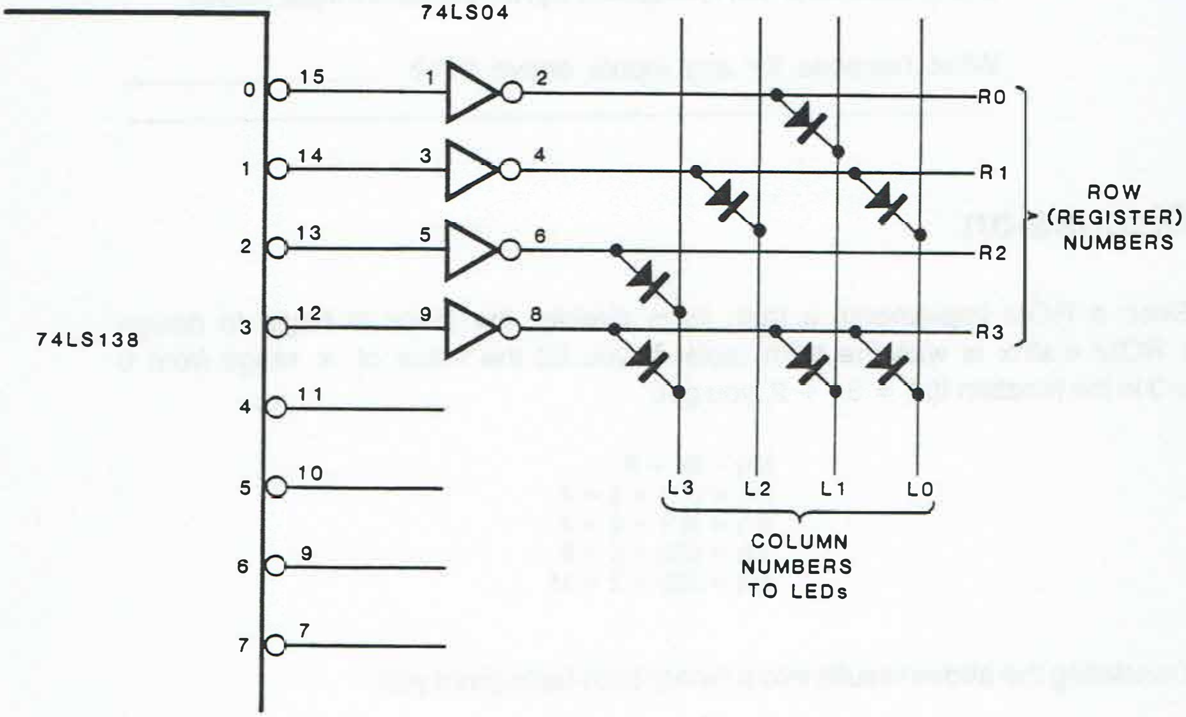


Figure E10-2

A diode ROM to implement f(x) = 3x + 2.

I have restricted the input value range, since you only have a limited number of diodes available. But the idea is the same regardless of the input range. In fact, this is exactly what you must do when ordering a commercial ROM. The manufacturer requires you to specify the binary or hex contents of each register in the ROM. You determine this directly from the problem truth table.

In summary, when using a ROM to implement a logic design, the ROM address lines become the circuit input lines and the ROM data lines are the circuit outputs. The contents of the internal ROM registers are taken directly from the problem truth table.

Procedure (continued)

5. Switch power OFF and remove the circuit components and wire.
6. Return to Unit 3 and complete the Unit Examination.

EXPERIMENT 11

REGISTER CIRCUITS

Objectives:

To verify the truth table of a D flip-flop.

To design and build a buffer storage register.

To demonstrate the use of a register control line.

To design and build a shift-left register.

To convert a shift-left register to a shift-right register.

Introduction

Register circuits generally come in two varieties: buffer registers and shift registers. A buffer register simply stores binary data, while a shift register shifts the stored bits left or right. The basic component of any register is the D flip-flop.

In this experiment, you will first verify the truth table of a D flip-flop. Then using D flip-flops, you will construct and demonstrate the operation of buffer and shift register circuits.

Materials Required

Heathkit Engineering Design Trainer ET-1000
1 — 74LS00 Quad NAND
1 — 74LS04 Hex Inverter
2 — 74LS74 Dual D flip-flop

Procedure

1. The 74LS74 has two internal D flip-flops as shown in Figure E11-1. Notice that the two flip-flops are totally independent with separate clock inputs. In addition, they include both the PRESET and CLEAR features. In the first part of this experiment, you will only use one of the flip-flops as shown in Figure E11-2.

   Connect the flip-flop to the Trainer as shown in Figure E11-2.

PIN 14 = +5V
PIN 7 = GROUND

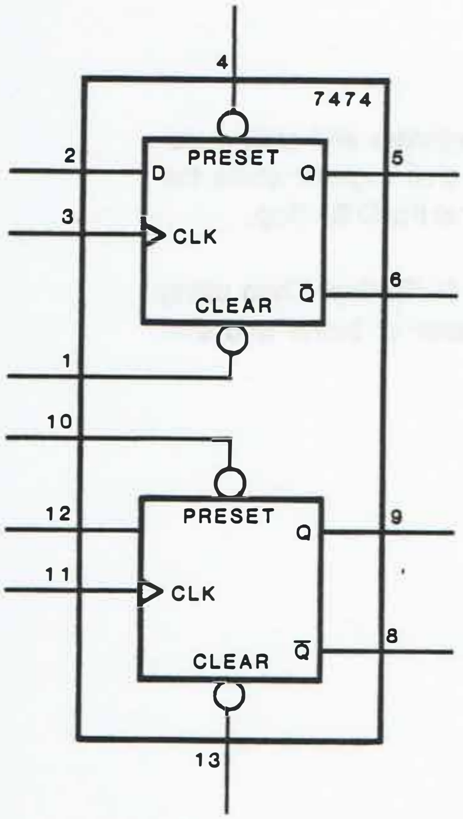
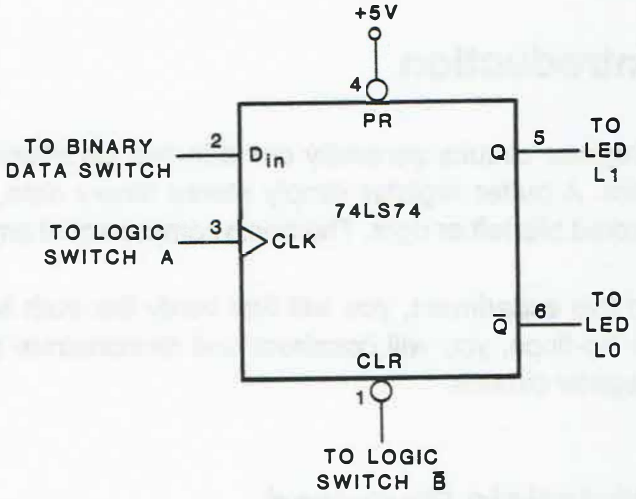


Figure E11-1
A pin diagram of the 74LS74 IC.

POWER: PIN 14 = +5V
PIN 7 = GROUND



CLK	D_{in}	Q	$\bar{Q}$	STATE
1	1			
1	0			
0	X			

Figure E11-2
A 74LS74 D flip-flop and truth table.

2. Set the binary data switch to its logic 0, or down, position. Then switch power ON and clear the flip-flop by pressing then releasing the spring-loaded logic switch labeled **B**. The Q output should clear and the $\overline{Q}$ output should set. Why? ______________________

   ____

3. Fill-in the truth table in Figure E11-2 as follows:

   SET state: Place the binary data switch connected to D_{in} in its logic 1, or up, position. Did the output change? Why?

   ____

   Apply a clock pulse by pressing then releasing the spring-loaded logic switch labeled **A**. Observe the output LEDs. The Q output should now be a logic 1, or set, and the $\overline{Q}$ output should be a logic 0, or cleared. When did the output change, when the clock switch was pressed or released? ______________________

   CLEAR state: Place the D_{in} binary data switch in its logic 0 position. Apply another clock pulse by pressing then releasing the **A** logic switch. Observe the output LEDs. The Q output should now be a logic 0, or cleared, and the $\overline{Q}$ output should be a logic 1, or set.

   HOLD state: Toggle the D_{in} binary data switch up and down between a logic 1 and a logic 0 state. **Do not** apply a clock pulse. Did the output change? Why? ______________________

Discussion

You have just demonstrated the operation of a 1-bit storage element, the D flip-flop. The flip-flop has three states: SET, CLEAR, and HOLD. The basic idea is that the input logic, D_{in}, is only transferred to the Q output when a clock pulse is applied. Otherwise, the output is held and remains unchanged from its previous state. In addition, the 74LS74 is clocked on a low-to-high clock transition. This is demonstrated by the fact that the flip-flop output only changes when the clock switch is pressed, not when it is released. You also observed that the Q output is always the opposite logic state of the $\overline{Q}$ output.

Now, let's combine the four D flip-flops in two 74LS74 ICs to produce a 4-bit buffer storage register.

Procedure (continued)

4. Switch power OFF and construct the 4-bit buffer register shown in Figure E11-3 using two 74LS74 ICs. Connect the four data input lines, D0 through D3, to binary data switches 0 through 3, respectively. Connect the four register output lines, Q0 through Q3, to LEDs L0 through L3, respectively.

   Notice in Figure E11-3 that each flip-flop is drawn as a separate logic device, much like a gate. In addition, the $\overline{Q}$ outputs are not shown. Many designers like to show all of the pins on a flip-flop, whether they are used or not. However, this isn't necessary, and often makes a schematic diagram too detailed and difficult to read. As a general rule, never show a pin if it isn't connected to the circuit, unless it is absolutely necessary to clarify a specific operation.

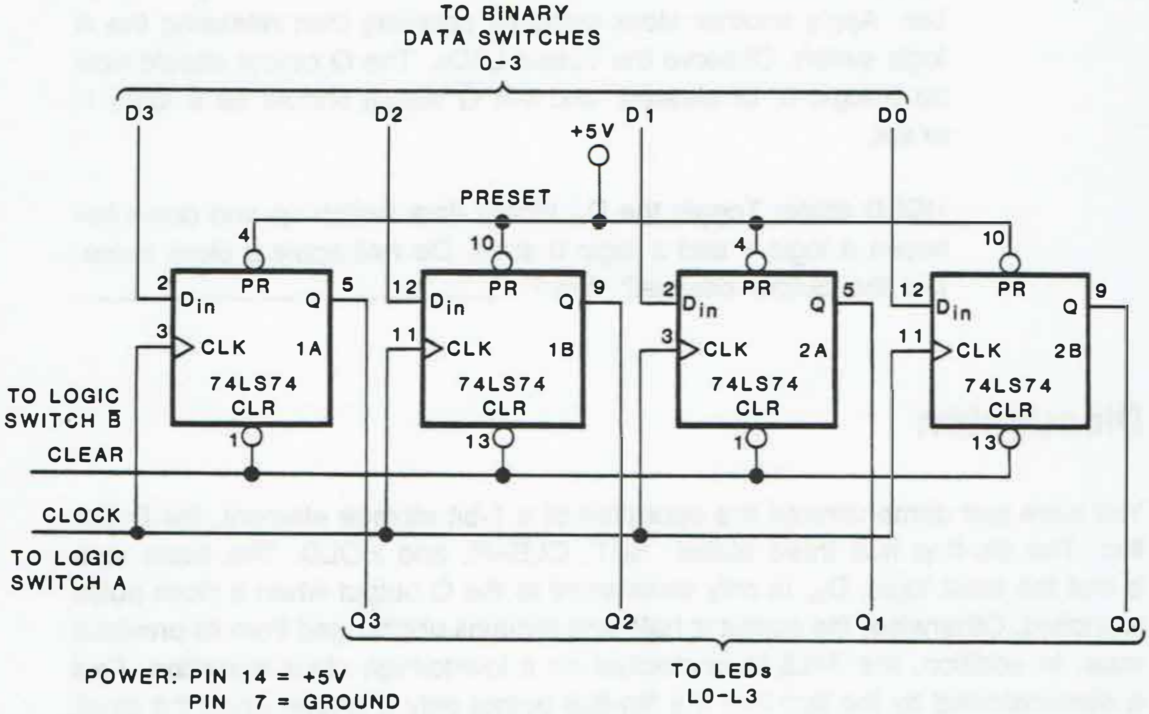


Figure E11-3

A 4-bit buffer register.

5. Switch power ON and clear the register by pressing then releasing the **B** logic switch.

6. Set the four binary data input switches to a random value.

7. Apply a clock pulse by pressing then releasing the **A** logic switch. Observe the output LEDs. They should indicate the binary input value. This value is now stored by the buffer register and it will not change until the input changes **and** another clock pulse is applied.

8. Change the data input to another random value. Did the register output change? ____________ Why? ____________________
____

9. Apply another clock pulse. The output should change to reflect the new input value.

Discussion

The 4-bit buffer register in Figure E11-3 is simply a collection of 1-bit flip-flop memory elements. The register stores, or latches, a binary input each time the clock line is activated. The 4-bit input word, **D**, appears on the output lines after the clock line is pulsed. In symbols, **Q** = **D** when CLK is active.

Procedure (continued)

10. In the Function Generator section of the Trainer, set the FREQ. control to 1 and the FREQ. MULTIPLIER switch to 1K.

11. Switch power OFF. Then reconnect the CLOCK line from logic switch A to the left (positive/negative transition) square wave output in the Function Generator section.

12. Switch power ON. Then set the four binary data switches in your circuit to a random value. What does the register output do?

    ____

    Why? ______________________________

13. Set the data input switches another random pattern and observe the register output. Does the output immediately indicate the input value? __________ If so, why? ____________________

    ____

Discussion

You have now applied a continuous 1 kHz clock signal to the register. As a result, you no longer have any control over when the data input is latched, since any change in the input value is immediately clocked to the output. To control the latching process you must add a **control line**.

Procedure (continued)

14. Switch power OFF and insert the gate network shown in Figure E11-4 between the binary data switches and the buffer register inputs. Notice that this gate network is used to control when logic switch data is applied to the register.

15. Switch power ON and set the data input switches to produce a random input value. Observe the register output. Does it reflect the input? __________ Why? ____________________

    ____

16. Activate the LOAD line by holding the **A** logic switch to the left. Does the register output reflect its input now? __________ Why?

    ____

    ____

17. Release the logic switch. Why does the output return to 0000?

    ____

18. Change the data input value and activate the LOAD line using the **A** logic switch. The register will transfer the data input to its output as long as the LOAD line is held high. When the LOAD line is inactive, the register output returns to 0000.

    How could the circuit be redesigned to HOLD the input value when the LOAD line is inactive? ______________________________

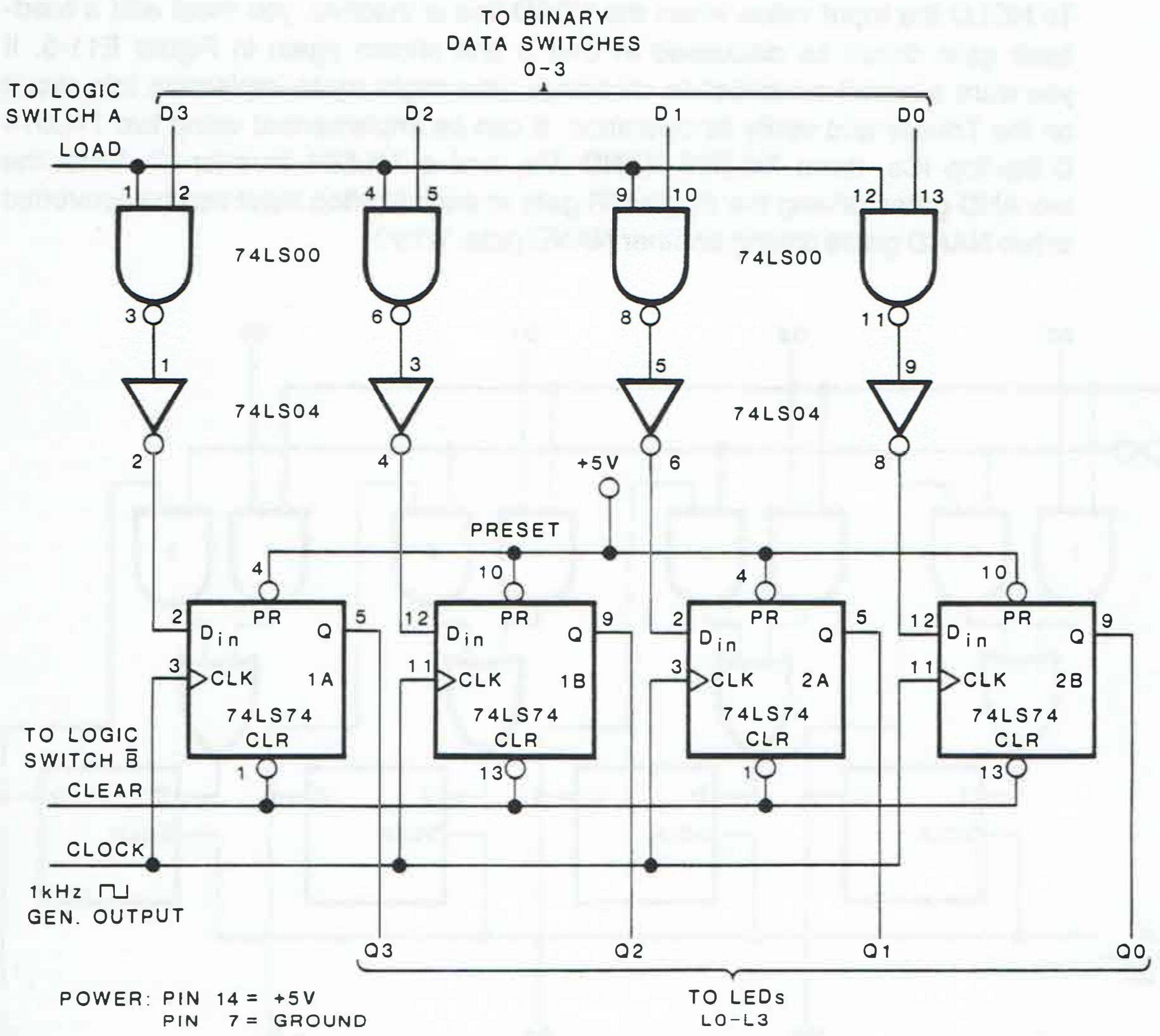


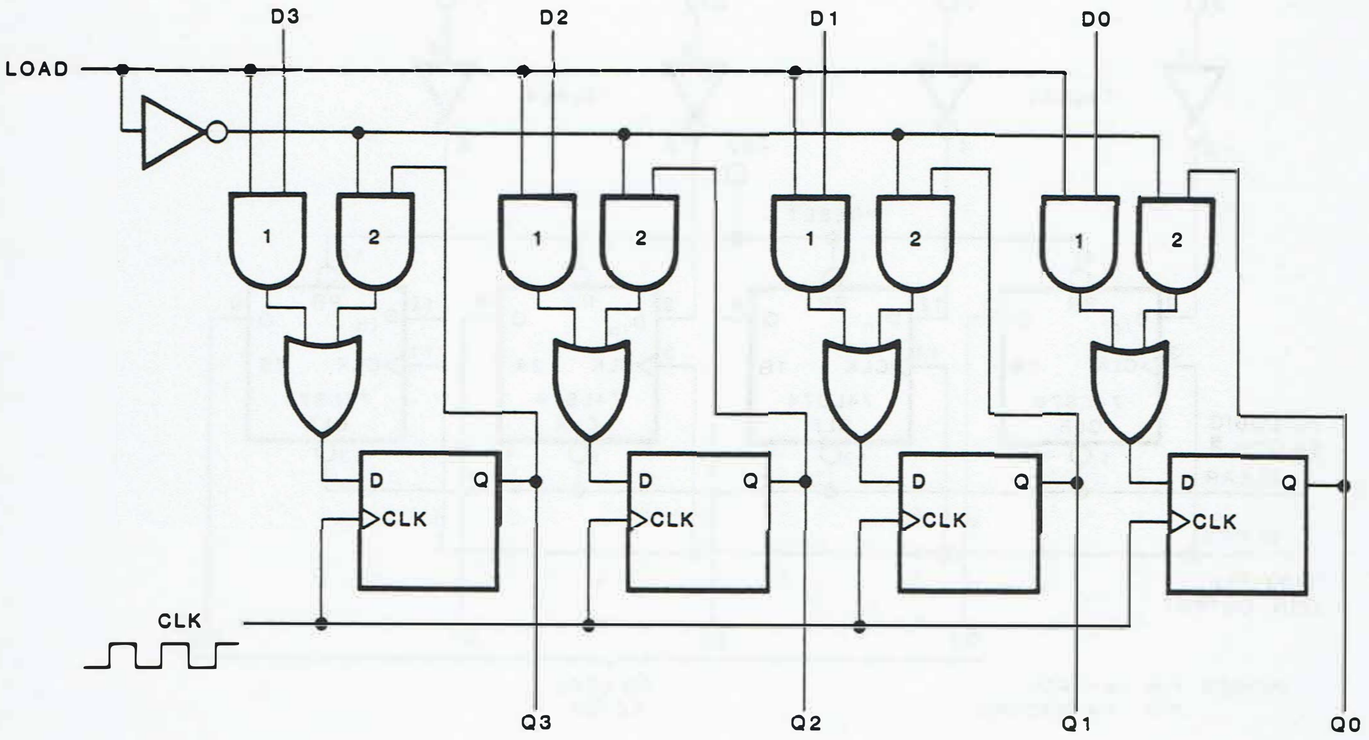
Figure E11-4

A 4-bit controlled buffer register.

Discussion

By adding the LOAD line, you can control precisely when an input value is to be latched while using a continuous clock signal. As a result, the register in Figure E11-4 is called a **controlled buffer register**. When the LOAD line is active, or high, the AND gates are enabled and the input data is passed to the register inputs. The next continuous clock pulse then latches the input value to the register output. However, when the LOAD line is inactive, or low, the gate structure applies logic 0's to the flip-flop inputs. As a result, the next clock pulse clears the register outputs.

To HOLD the input value when the LOAD line is inactive, you must add a **feedback** gate circuit as discussed in Unit 3 and shown again in Figure E11-5. If you want a circuit construction challenge, you might try to implement this circuit on the Trainer and verify its operation. It can be implemented using two 74LS74 D flip-flop ICs, three 74LS00 NAND ICs, and a 74LS04 Inverter IC. Note: the two AND gates driving the single OR gate at each flip-flop input can be converted to two NAND gates driving another NAND gate. Why?



LOAD	D_n	Q_n	STATE
0	X	LAST	HOLD
1	0	0	LATCH
1	1	1	LATCH

Figure E11-5

A 4-bit controlled buffer register with feedback.

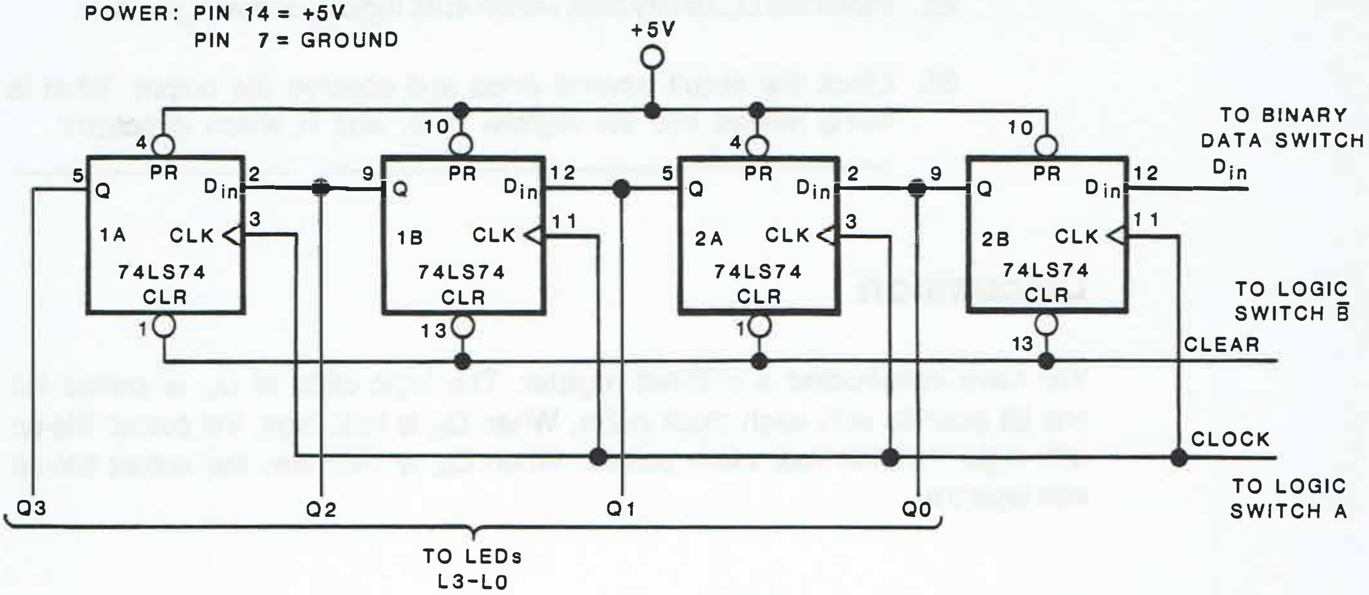


Figure E11-6
A 4-bit shift-left register.

Procedure (continued)

19. Switch power OFF and construct the 4-bit shift-left register shown in Figure E11-6. I recommend that you tear-down your previous circuit and reconstruct this one from scratch, since it is quite different.

20. Switch power ON and clear the register with the **B** logic switch.

21. Switch the D_{in} binary data switch to its logic 1, or up, position.

22. Clock the circuit with the **A** logic switch and observe the output. You should see a logic 1 shifted into the least significant bit position.

23. Clock the circuit again and observe the output. You should now see logic 1's in the two least significant bit positions.

24. Clock the circuit until you get an output of 1111. How many more clock pulses did this take? ____________ What happens when more clock pulses are applied? ______________________

____

25. Place the D_{in} binary data switch in its logic 0, or down, position.

26. Clock the circuit several times and observe the output. What is being shifted into the register now, and in which direction?

____

Discussion

You have constructed a shift-left register. The logic state of D_{in} is shifted left one bit position with each clock pulse. When D_{in} is held high, the output fills-up with logic 1's after four clock pulses. When D_{in} is held low, the output fills-up with logic 0's.

Procedure (continued)

27. Switch power OFF and reverse the register output lines. That is, swap Q0 with Q3 and Q1 with Q2. This is done by simply changing the LED connections.

    How will this change affect the function of the circuit? ______

    ____

28. Switch the power ON and clear the register.

29. Put the D_{in} binary data switch in its logic 1 position.

30. Clock the circuit several times while observing the output. What is the register doing? ______________________

    ____

31. Put the D_{in} binary data switch in its logic 0 position and clock the circuit several times. What is the register doing now? ______

    ____

Discussion

By reversing the register outputs, you have created a shift-right register. When D_{in} is held high, consecutive clock pulses will now shift a logic 1 to the right until the entire output is filled-up with 1's. Conversely, when D_{in} is held low, logic 0's are shifted right.

Procedure (continued)

32. Switch power OFF and remove the circuit components and wire.

33. Proceed to Experiment 12.

EXPERIMENT 12

COUNTER CIRCUITS

Objectives:

To verify the truth table of a JK flip-flop.

To construct and demonstrate the operation of a 4-bit ripple counter.

To modify a standard 4-bit ripple counter to produce a BCD count.

To demonstrate the use of a control line to control the operation of a counter.

To modify a standard 4-bit ripple counter to create a counter of any modulus.

To investigate the operation of MSI counters.

Introduction

One of the most useful circuits in digital electronics is the counter. A counter simply counts clock pulses. Thus, the input to the counter is a series of clock pulses and the output is a multi-bit binary count value.

Most counters are made of JK flip-flops operated in their toggling mode. These counters can be made to count up or count down. In addition, they can be wired to generate any number of unique counts within a given count range. The number of unique count values generated by a counter is called its **modulus**. Thus, a BCD counter that counts from 0000 to 1001 is called a mod 10 counter.

In this experiment, you will first verify the operation of a JK flip-flop. Next, you will use four JK flip-flops to construct a 4-bit binary ripple counter. Then you will modify the counter to generate a BCD count as well as any modulus between 1 and 16. Finally, you will experiment with two popular MSI counter ICs.

Materials Required

Heathkit Engineering Design Trainer ET-1000
1 — 74LS02 Quad 2-bit NOR
1 — 74LS20 Dual 4-bit NAND
2 — 74LS76 Dual JK Flip-Flop
1 — 74LS90 Decade Counter
1 — 74LS193 Binary Up/Down Counter

Procedure

1. A pin diagram of the 74LS76 IC is shown in Figure E12-1. Notice that it contains two separate edge triggered JK flip-flops. In the first part of this experiment, you will only use one of the flip-flops as shown in Figure E12-2.

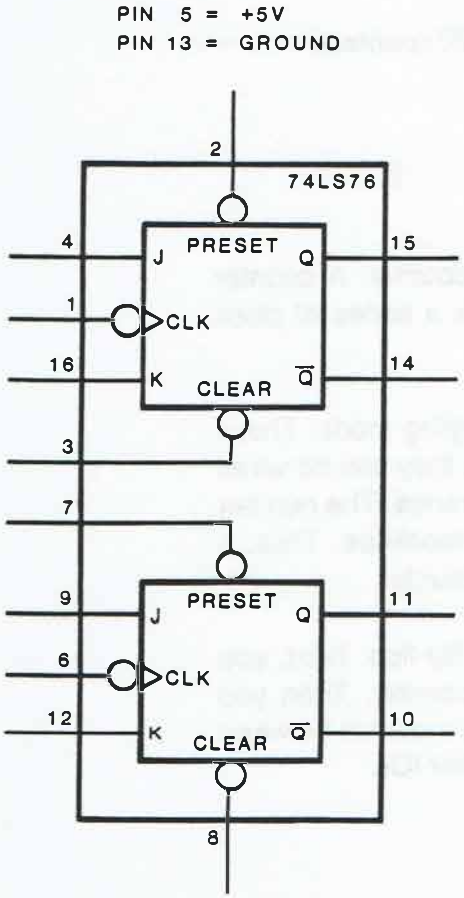
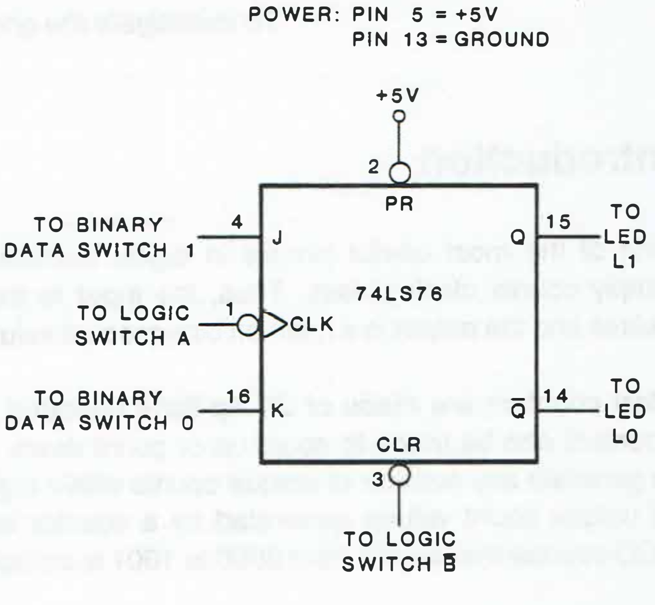


Figure E12-1
A pin diagram of the 74LS76 IC.



J	K	Q	Q̄	STATE
1	0			
0	0			
0	1			
1	1			

Figure E12-2
A 74LS76 JK flip-flop and truth table.

Connect the flip-flop shown in Figure E12-2 to the Trainer. Notice that to power the IC, you must connect +5 volts to pin 5 and ground to pin 13.

2. Switch power ON and CLEAR the flip-flop by pressing then releasing the **B** logic switch. The Q output should clear and the $\overline{Q}$ output should set.

3. Fill-in the truth table in Figure E12-2 as follows:

SET state: Set the J and K binary data switches to produce an input of JK = 10. Apply a clock pulse by pressing then releasing the **A** logic switch and observe the output LEDs. The Q output should set and the $\overline{Q}$ output should clear.

HOLD state: Set the J and K binary data switches to produce an input of JK = 00. Apply several clock pulses and notice that the outputs are held in the previous set state.

CLEAR state: Set the J and K binary data switches to produce an input of JK = 01. Apply a clock pulse and observe the LEDs. This time, the Q output should clear and the $\overline{Q}$ output should set.

TOGGLE state: Set the J and K inputs to produce an input of JK = 11. Apply several clock pulses and notice that the outputs toggle from 0-to-1-to-0-to-1, and so on, for each clock pulse.

When does the output change, when the logic switch is pressed or released? ____________ What does this tell you about the way the 74LS76 is triggered? ______________________

Discussion

A JK flip-flop generates all the output states available from a D flip-flop and, in addition, includes a TOGGLE state. The TOGGLE state is particularly useful for constructing counter circuits. Once the flip-flop inputs are set to produce a given state, the outputs will not change until a clock pulse is applied. The 74LS76 requires a high-to-low transition for clocking. Consequently, this device is negative edge triggered. You should have noticed that the outputs did not change until the spring-loaded clock switch was released to produce a negative edge.

Finally, the 74LS76 also has PRESET and CLEAR inputs. You disabled the PRESET input by tying it high. The CLEAR input was connected to a logic switch so that the Q output could be cleared without applying a clock pulse.

Procedure (continued)

4. Switch power OFF and construct the 4-bit ripple counter circuit shown in Figure E12-3 using two 74LS76 ICs.

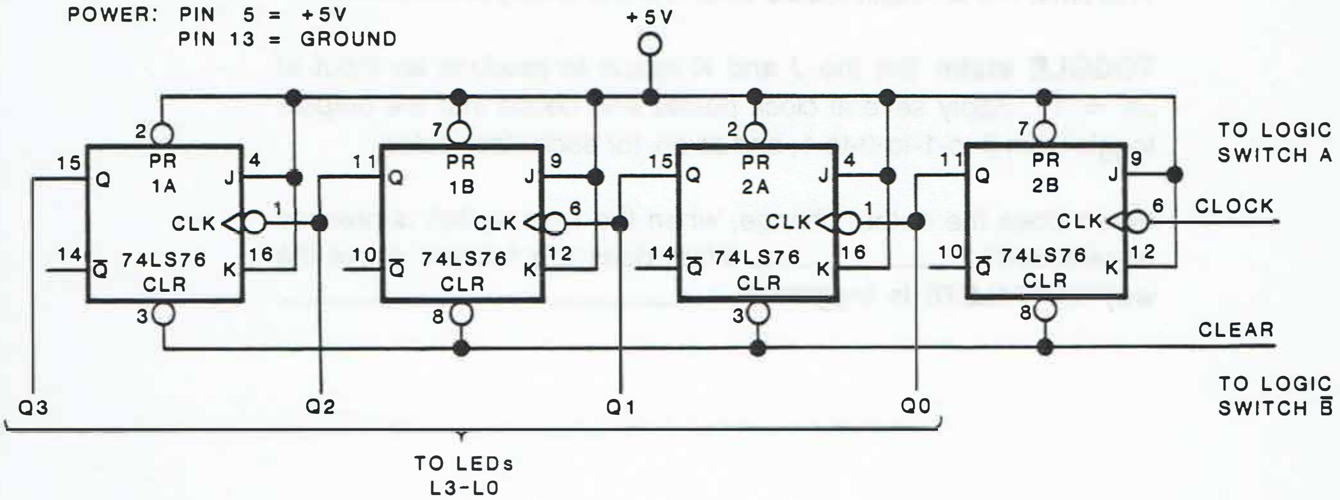


Figure E12-3

A 4-bit ripple counter.

5. Switch power ON and clear the counter by pressing then releasing the **B** logic switch.

6. Fill-in the following state table by applying 16 successive clock pulses using the **A** logic switch.

State	Q3	Q2	Q1	Q0	
0	0	0	0	0	(CLEAR)
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
0					

7. Switch power OFF and reconnect the wires coming from the flip-flop Q outputs (pins 11 and 15 on each IC) to the $\overline{Q}$ outputs (pins 10 and 14 on each IC).

8. Switch power ON and clear the counter. Why do the output LEDs indicate 1111 rather than 0000? ____________________

____

9. Fill-in the following state table by applying 16 successive clock pulses using the **A** logic switch. What is the counter doing now?

State	$\overline{Q3}$	$\overline{Q2}$	$\overline{Q1}$	$\overline{Q0}$	
0	1	1	1	1	(CLEAR)
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
0					

Discussion

The ripple counter gets its name from the idea that the output of one flip-flop clocks the next flip-flop, and so on, to create a ripple effect through the counter. All of the flip-flops must be placed in their toggling mode by tying their J and K inputs high.

As the circuit is clocked, the Q outputs count up and the $\overline{Q}$ outputs count down. This is shown by the two preceding state tables. Larger counts are possible by simply adding more flip-flop stages to the counter.

There are sixteen unique counts for the 4-bit ripple counter, from 0000 to 1111. Thus, the modulus of the counter is 16. The maximum count value, however, is one less than the counter modulus, or 15.

Procedure (continued)

10. Switch power OFF and modify the 4-bit ripple counter as shown in the circuit diagram in Figure E12-4. Notice that:

    A. All of the J and K inputs are now tied together and connected to a binary data switch. This is the counter COUNT line.

    B. The PR inputs of the flip-flops are still connected to +5 volts.

    C. The CLR inputs of the flip-flops are now connected to the output of a NOR gate. The NOR gate, in turn, receives the CLEAR signal from the **B** logic switch and the ANDed outputs from the four flip-flops. **Do not** connect the inputs to the NAND gate at this time. The counter modulus will determine which Q or $\overline{Q}$ outputs will be used.

    D. The CLK input of the first (least significant) flip-flop is connected to the left (positive/negative transition) square wave output in the Function Generator section of the Trainer.

    E. The Q outputs of the counter are connected to the LEDs and **NOT** the $\overline{Q}$ outputs.

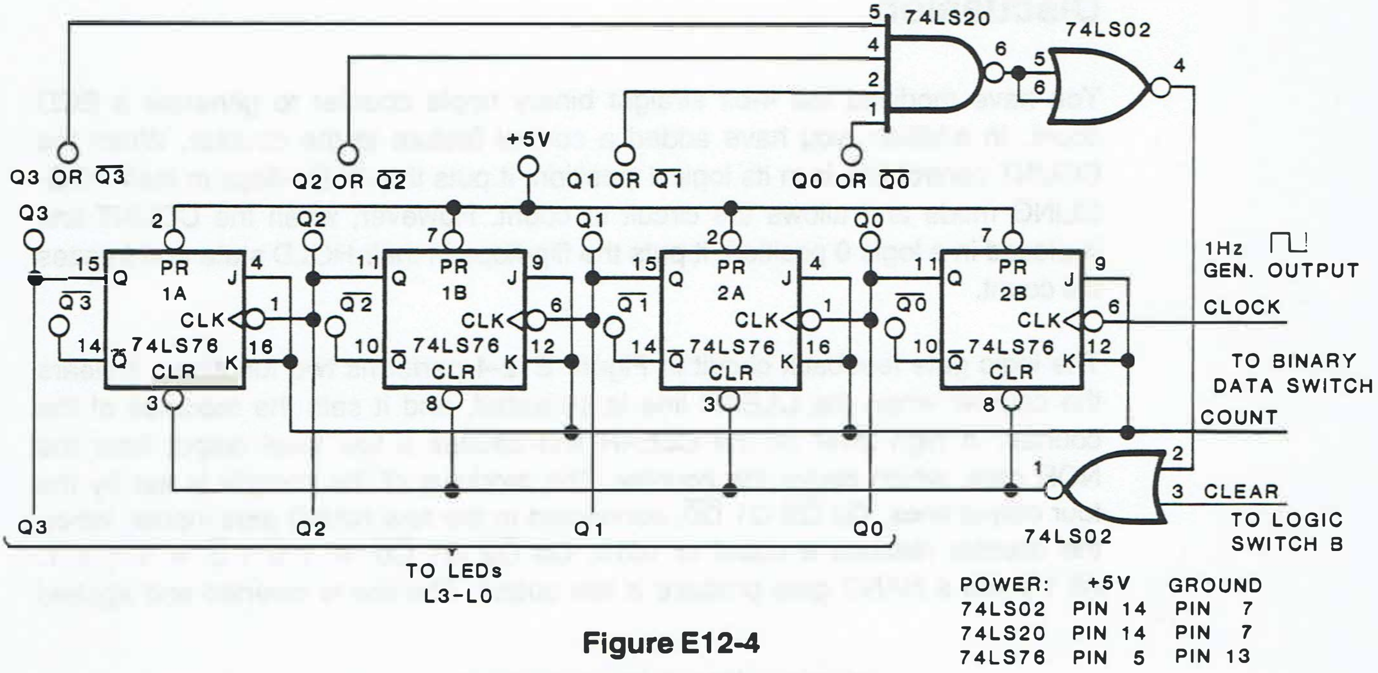


Figure E12-4

A controlled, variable modulus counter.

11. Set both the FREQ. control and FREQ. MULTIPLIER switch in the Function Generator section of the Trainer to 1.

12. Connect the four inputs of the 4-bit NAND gate to the Q3, $\overline{Q2}$, Q1, and $\overline{Q0}$ outputs of the counter.

13. Place the COUNT (J and K inputs) binary data switch in its logic 1, or up, position.

14. Switch power ON an observe the counter output. You should see a BCD count from 0000 to 1001, that repeats the sequence after 1001.

15. Press then release the **B** logic switch a number of times, and observe that you can clear the counter any time during the count sequence.

16. While the circuit is counting, place the COUNT binary data switch in its logic 0, or down, position. What happens and why?

    ____

17. Place the COUNT binary data switch in its logic 1, or up, position. The circuit will resume counting.

Discussion

You have modified the 4-bit straight binary ripple counter to generate a BCD count. In addition, you have added a control feature to the counter. When the COUNT control line is in its logic 1 position, it puts the JK flip-flops in their TOGGLING mode and allows the circuit to count. However, when the COUNT line is placed in a logic 0 position, it puts the flip-flops in their HOLD state and freezes the count.

The logic gate feedback circuit in Figure E12-4 performs two functions: it clears the counter when the CLEAR line is activated, and it sets the modulus of the counter. A high level on the CLEAR line causes a low level output from the NOR gate, which clears the counter. The modulus of the counter is set by the four output lines, Q3 $\overline{Q2}$ Q1 $\overline{Q0}$, connected to the four NAND gate inputs. When the counter reaches a count of 1010, Q3 $\overline{Q2}$ Q1 $\overline{Q0}$ = 1 $\overline{0}$ 1 $\overline{0}$ = 1 1 1 1. All 1's into a NAND gate produce a low output. The low is inverted and applied to the other side of the "CLEAR line" NOR gate. This causes a low output from the NOR gate to clear the counter. In other words, the counter counts up to 1001, then rolls over to 0000 as soon as the feedback circuit detects a 1010 output. The result is a mod 10 decade, or BCD, counter.

How do you suppose you could change the counter modulus? To change the counter modulus, you simply change the flip-flop connections to the feedback circuit. The modulus of your counter is 10, in decimal. If you convert that to binary, you get 1010. This translates to a fundamental product output of Q3 $\overline{Q2}$ Q1 $\overline{Q0}$. Now suppose you want to make a mod 5 counter. The 4-bit binary equivalent of 5 is 0101. This translates to a fundamental product output of $\overline{Q3}$ Q2 $\overline{Q1}$ Q0 from the counter. So, to create a mod 5 counter, you must connect $\overline{Q3}$, Q2, $\overline{Q1}$, and Q0 to the NAND gate input lines.

Procedure (continued)

18. Switch power OFF and make a mod 5 counter out of your circuit by connecting the NAND gate inputs to counter outputs $\overline{Q3}$, Q2, $\overline{Q1}$, and Q0.

19. Switch power ON and verify that your counter is a mod 5 counter and counts from 0000 to 0100.

20. Modify your circuit to generate several different moduli by changing the Q and $\overline{Q}$ connections to the feedback circuit. Verify your modifications by operating the counter.

Discussion

The counter can be made to generate any modulus between 1 and 15 by changing the Q and $\overline{Q}$ inputs to the feedback circuit. How can the counter be made into mod 16 counter, to count from 0000 to 1111? (Hint: What must the NAND gate inputs be in order to keep the feedback circuit from clearing the counter?)

To keep the feedback circuit from clearing the counter, the NOR gate output must be held high, which means the NAND gate inputs must be held low. So, in order to generate a straight binary count from 0000 to 1111 (mod 16) the NAND gate inputs must be hardwired to ground. Try it!

Procedure (continued)

21. Switch power OFF and remove the circuit components and wire. Then construct the 74LS90 decade counter circuit shown in Figure E12-5.

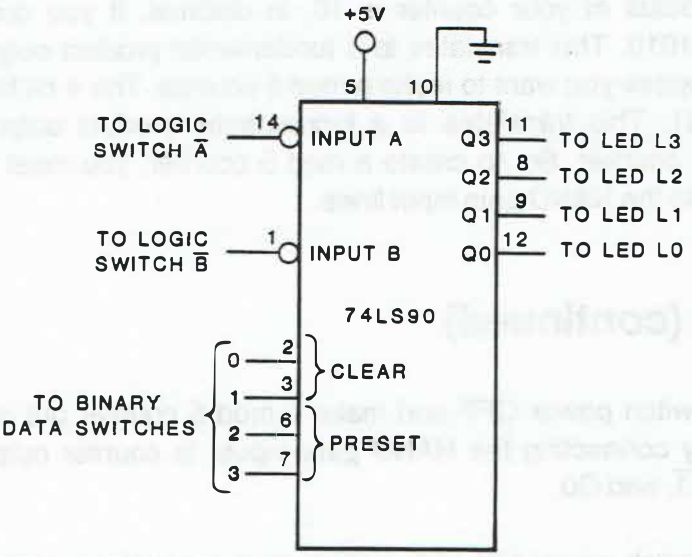


Figure E12-5

An MSI decade counter, the 74LS90.

22. Switch power ON and observe the LEDs.

23. Preset the counter by placing binary data switches 2 and 3 in their logic 1, or up, position. What value is preset into the counter?

24. Return the PRESET binary data switches to their logic 0, or down, position.

25. Clear the counter by placing binary data switches 0 and 1 in their logic 1, or up, position. Are all the LEDs indicating a logic 0?

26. Return the CLEAR binary data switches to their logic 0, or down, position.

27. Actuate the INPUT **A** logic switch several times and observe the counter output. What is it doing? ____________
____

28. Clear the counter, actuate the INPUT **B** logic switch several times, and observe the counter output. What is it doing now?
____

29. Switch power OFF. Then disconnect the INPUT B line (pin 1) from the **B** logic switch and connect it to the Q0 output line (pin 12). Switch power ON.

30. Actuate the INPUT **A** logic switch at least ten times to sequence the counter through all of its output states. What is the counter doing now? ____________
____

Discussion

The 74LS90 is a divide-by-10, or decade, counter. It contains a divide-by-2 section and a divide-by-5 section. Applying a clock pulse to the INPUT A line of the counter causes the Q0 output to toggle and generate a divide-by-two count. Applying a clock pulse to the INPUT B line generates a divide-by-5 count on output lines Q3, Q2, and Q1. You should have observed that Q3, Q2, and Q1 counted from 000 to 100 in step 28.

A divide-by-10 counter is created by connecting the Q0 output of the divide-by-2 section to the INPUT B line of the divide-by-5 section. Applying a clock pulse to the INPUT A line then results in a BCD, or decade, count.

The counter can also be preset to 1001 by applying logic 1's to the PRESET inputs (pins 6 and 7). The counter is cleared to 0000 by placing logic 1's on the CLEAR inputs (pins 2 and 3). The PRESET and CLEAR inputs must all be low for the counter to count.

Procedure (continued)

31. Switch power OFF and remove the 74LS90 IC and wires. Then construct the circuit shown in Figure E12-6. Note that pin 5 is connected to the left (positive/negative transition) square wave output in the Function Generator section of the Trainer, and the rate is set for 1 Hz.

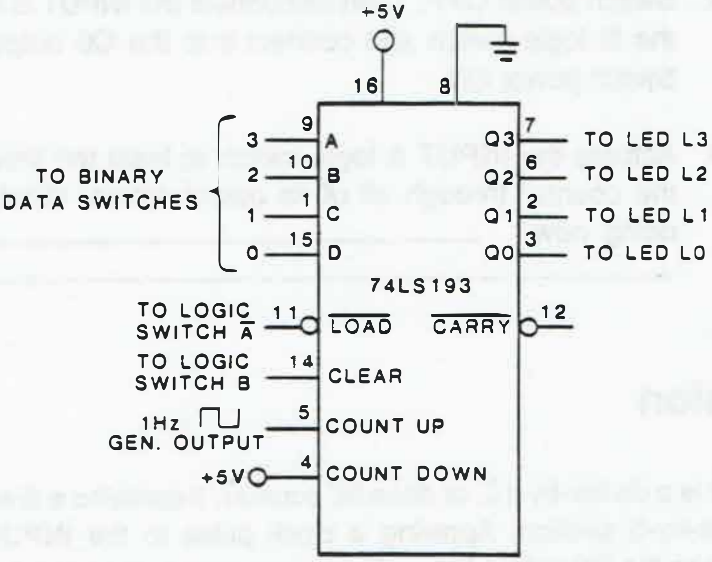


Figure E12-6
A 4-bit binary up/down counter.

32. Switch power ON and observe the counter output. What type of counter is this? ____________________

33. You can clear the count at any time by actuating the CLEAR input via logic switch **B**. Try it!

34. Set the four binary data switches to produce an ABCD input value of 0101.

35. Actuate the $\overline{\text{LOAD}}$ line and observe the counter output. Do it again! What is the purpose of the $\overline{\text{LOAD}}$ input? ____________________

____

36. Switch power OFF. Then swap the wires at pins 4 and 5. The square wave signal should now be connected to pin 4 and +5 volts to pin 5. Switch power ON and observe the counter output. What type of counter is it now? ____________________
____

Discussion

The 74LS193 is a 4-bit binary up/down counter. The counter counts-up when a clock pulse is applied to the COUNT UP input and counts-down when a clock pulse is applied to the COUNT DOWN input. The counter can be preset to begin counting at any value via the $\overline{\text{LOAD}}$ input control line. When the $\overline{\text{LOAD}}$ line is activated, or low, the binary value at the ABCD inputs is loaded into the counter. The counter begins counting with the loaded value, then rolls over to 0000 unless the $\overline{\text{LOAD}}$ line is activated again. Note that the ABCD inputs are reversed from those specified by the manufacturer, to maintain consistency between the text and the experiment.

Procedure (continued)

37. Switch power OFF and again swap the wires at pins 4 and 5.

38. Disconnect the $\overline{\text{LOAD}}$ input from logic switch **A** and connect it to the $\overline{\text{CARRY}}$ input at pin 12.

39. Switch power ON and observe the count sequence. How does this differ from before? ____________________
____

Discussion

The 74LS193 has a $\overline{\text{CARRY}}$ output that is activated each time the counter attempts to roll over from 1111 to 0000. Since you have connected this output line to the $\overline{\text{LOAD}}$ input, the counter rolls over to the ABCD preset input value. As a result, you have created a presettable counter that can be preset with any value between 0000 and 1111.

Procedure (continued)

40. Change the ABCD preset inputs to any random value and observe that the counter begins counting at the preset value.

41. Switch power OFF and remove the IC and wires.

42. Return to Unit 4 and read the section on “Designing Sequential Circuits.”

EXPERIMENT 13

SEQUENTIAL CIRCUIT DESIGN

Objectives:

To demonstrate the sequential circuit design process.

To implement a sequential circuit design using D flip-flops.

To implement a sequential circuit design using JK flip-flops.

Introduction

Sequential circuits that are designed to generate unique codes and sequences often require combinational logic to process input and output signals for the flip-flop memory elements. In most cases, the combinational logic forms a feedback circuit which samples the circuit outputs and sets-up the flip-flop inputs for the next clock pulse.

In Unit 4, you were acquainted with a step-by-step procedure that allowed you to design sequential circuits to generate any required binary code sequence. This sequential circuit design algorithm is summarized by the flowchart in Figure E13-1. Using this procedure, the sequential circuit can be designed using either D or JK flip-flops. However, the JK design usually results in less combinational feedback logic.

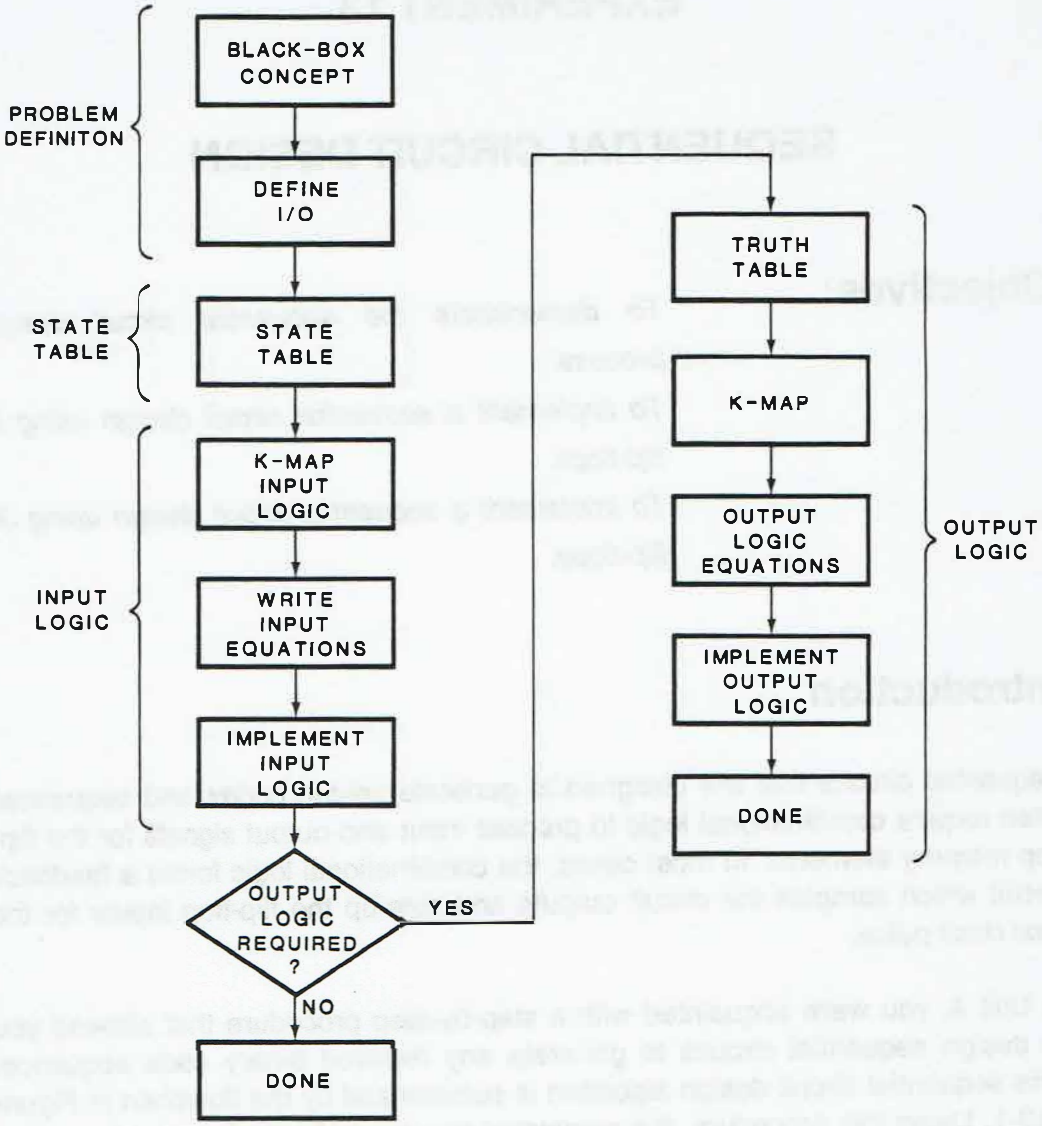


Figure E13-1

A flowchart summary of the sequential circuit design algorithm.

In this experiment, you will be given a binary code sequence and design both D and JK flip-flop circuits to generate the sequence. Of course, you will also implement the circuits on the Trainer to verify that they do, indeed, satisfy the design requirements.

Materials Required

Heathkit Engineering Design Trainer ET-1000
1 — 74LS00 Quad 2-bit NAND
1 — 74LS02 Quad 2-bit NOR
2 — 74LS74 Dual D flip-flop
2 — 74LS76 Dual JK flip-flop
1 — 74LS86 Quad 2-bit XOR

Procedure

1. An industrial process requires a sequential control circuit to implement the following logic sequence: 000, 010, 101, 001, 011.

   Apply the sequential design algorithm to design a D flip-flop control circuit that will implement the above sequence. If you have trouble, refer to the following discussion.

2. Construct your circuit on the Trainer and verify that it generates the required sequence.

Discussion

You should obtain the following results by applying the sequential design algorithm.

PROBLEM DEFINITION

The problem definition phase of the design process requires a black-box design concept as shown in Figure E13-2. Since the circuit simply counts the given sequence, the input to the circuit is a clock signal whose frequency is determined by the application. I have used the **A** logic switch on the Trainer to supply the clock signal. In addition to the clock input, a CLEAR line is included to initialize the circuit to its initial state.

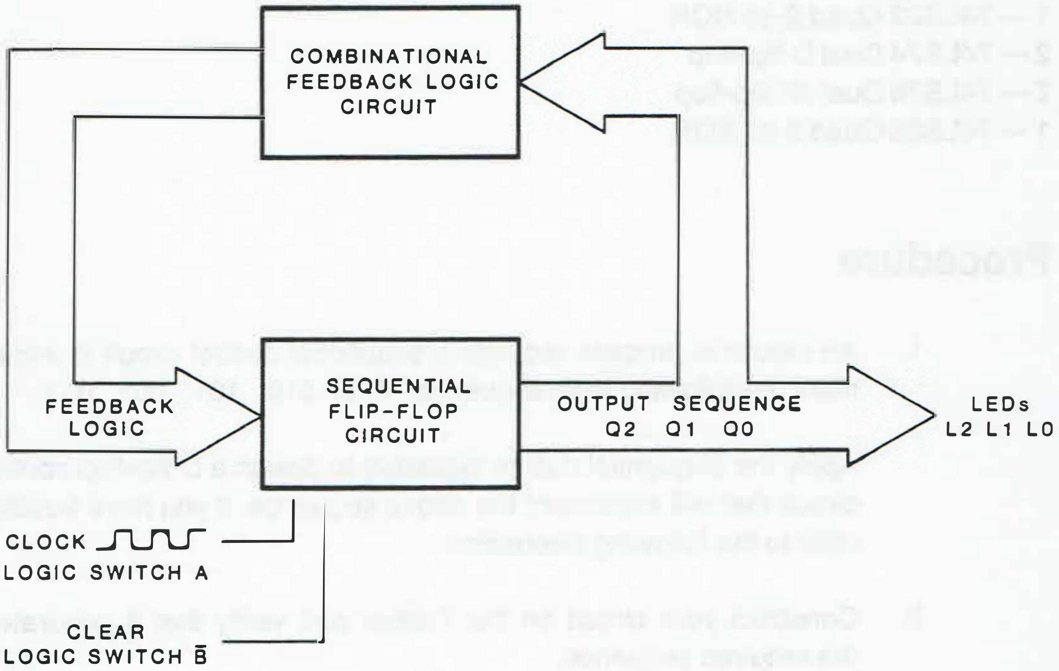


Figure E13-2

A black-box concept for the sequential circuit design problem.

The black-box concept shows that combinational feedback logic must be designed to sample a given output and generate the necessary flip-flop input logic to prepare the circuit for the next count. The sequential output sequence is generated by the Q2, Q1, and Q0 outputs of three flip-flops. The flip-flop outputs will be connected to LEDs L2, L1, and L0 on the Trainer, respectively, to indicate the sequential states.

STATE TABLE

The state table must show all the legal and illegal output states of the circuit. In addition, the state table must show both the **t** and **t + 1** states as follows:

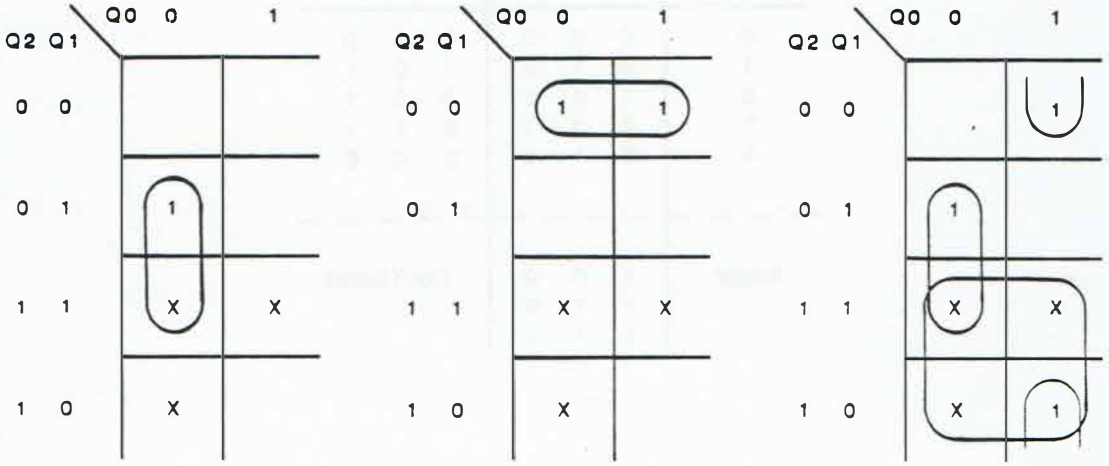
	t			**t + 1**		
State	Q2	Q1	Q0	Q2	Q1	Q0
0	0	0	0	0	1	0
1	0	1	0	1	0	1
2	1	0	1	0	0	1
3	0	0	1	0	1	1
4	0	1	1	0	0	0
Illegal	1	0	0	Don't Cares		
	1	1	0			
	1	1	1			

THE INPUT LOGIC

A logic 1 must be applied to a given flip-flop input when its next, or t+1, state must be high. Thus, you must locate the t states where a given output is high in the next, or t+1, state. These t states must then be plotted in a K-map for each flip-flop input as shown in Figure E13-3. Notice that there is a K-map for each flip-flop input: D2, D1, and D0. The K-map grids use the three circuit outputs (Q2, Q1, and Q0) as the grid variables. The illegal states are plotted as don't cares to aid in the simplification process.

Once each K-map is plotted, the plots are grouped and read to produce the following logic equations:

$$D2 = Q1\,\overline{Q0}$$
$$D1 = \overline{Q2}\,\overline{Q1}$$
$$D0 = Q2 + Q1\,\overline{Q0} + \overline{Q1}\,Q0$$



Q2 Q1 \ Q0	0	1
0 0		
0 1	1	
1 1	X	X
1 0	X	

Q2 Q1 \ Q0	0	1
0 0	1	1
0 1		
1 1	X	X
1 0	X	

Q2 Q1 \ Q0	0	1
0 0		1
0 1	1	
1 1	X	X
1 0	X	1

$$D2 = Q1\,\overline{Q0} = \overline{\overline{Q1}+Q0}$$

$$D1 = \overline{Q2}\;\overline{Q1} = \overline{Q2+Q1}$$

$$D0 = Q2 + Q1\,\overline{Q0} + \overline{Q1}\,Q0 = Q2 + (Q0 \oplus Q1)$$

STATE	t	t + 1
0	0 0 0	0 1 0
1	0 1 0	1 0 1
2	1 0 1	0 0 1
3	0 0 1	0 1 1
4	0 1 1	0 0 0
	1 0 0	X
	1 1 0	X
	1 1 1	X

Figure E13-3

State table and K-maps for the D flip-flop input logic.

The final step is to implement the feedback logic. Converting the AND/OR logic to NOR/XOR logic you get the circuit shown in Figure E13-4. Here, D2 and D1 have been converted to NOR logic, while D0 has been converted to NOR/XOR logic. Notice that the last two product terms in the D0 equation reduce to an XOR operation, since $Q1\,\overline{Q0} + \overline{Q0}\,Q1 = Q0 \oplus Q1$.

The circuit in Figure E13-4 can be implemented with four ICs: a 74LS02 NOR IC, a 74LS86 XOR IC, and two 74LS74 D flip-flop ICs. Notice that we used a different method to designate the D input and Q output for each flip-flop. Rather than a single letter, as before, we used a letter along with a number to identify the specific input or output. Thus, D2 is the input for the third flip-flop and Q0 is the output for the first flip-flop. You can use either method when when you draw your circuit diagram.

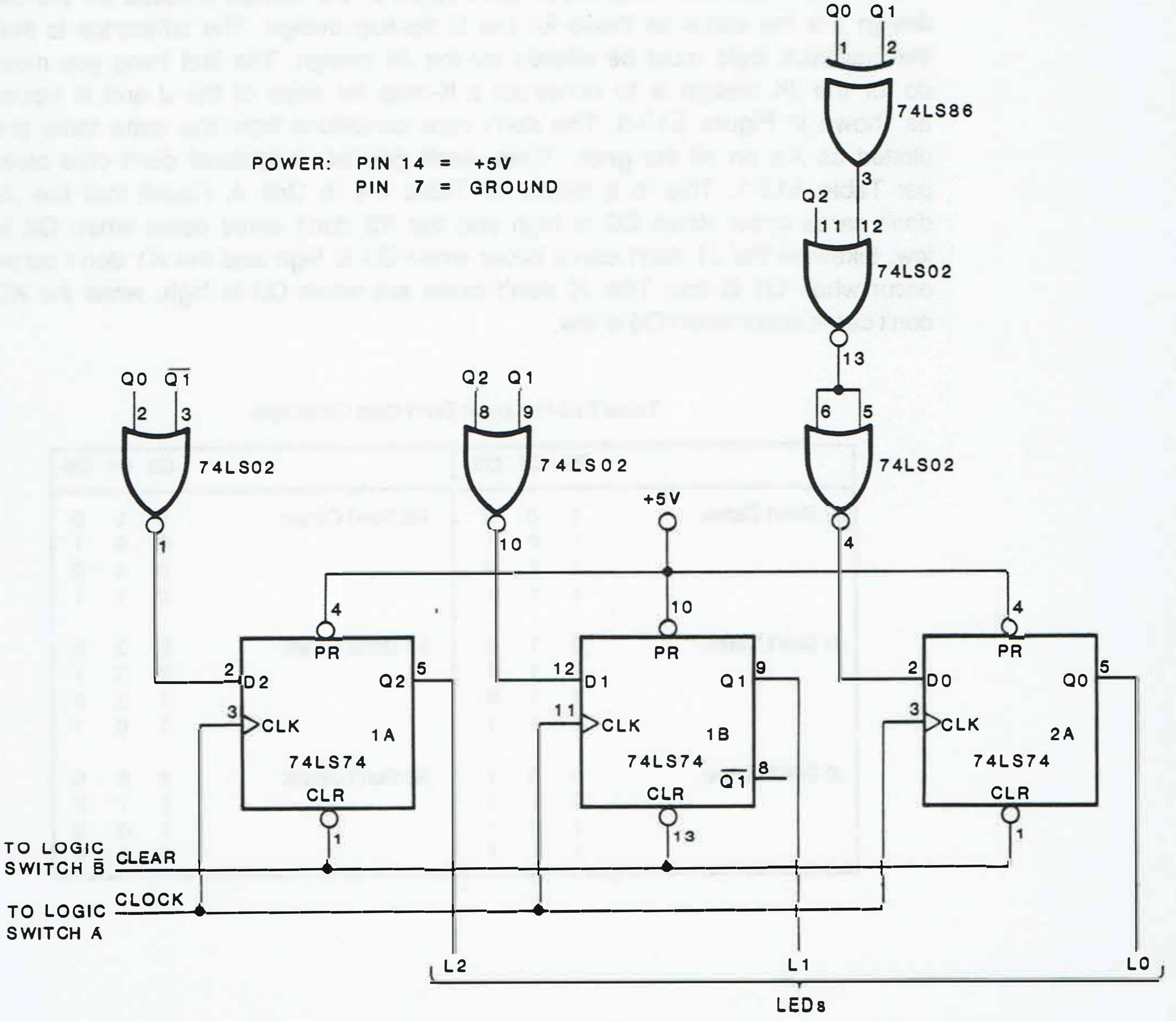


Figure E13-4

The final D flip-flop design.

Procedure (continued)

3. Design a JK flip-flop circuit to implement the preceding logic sequence.

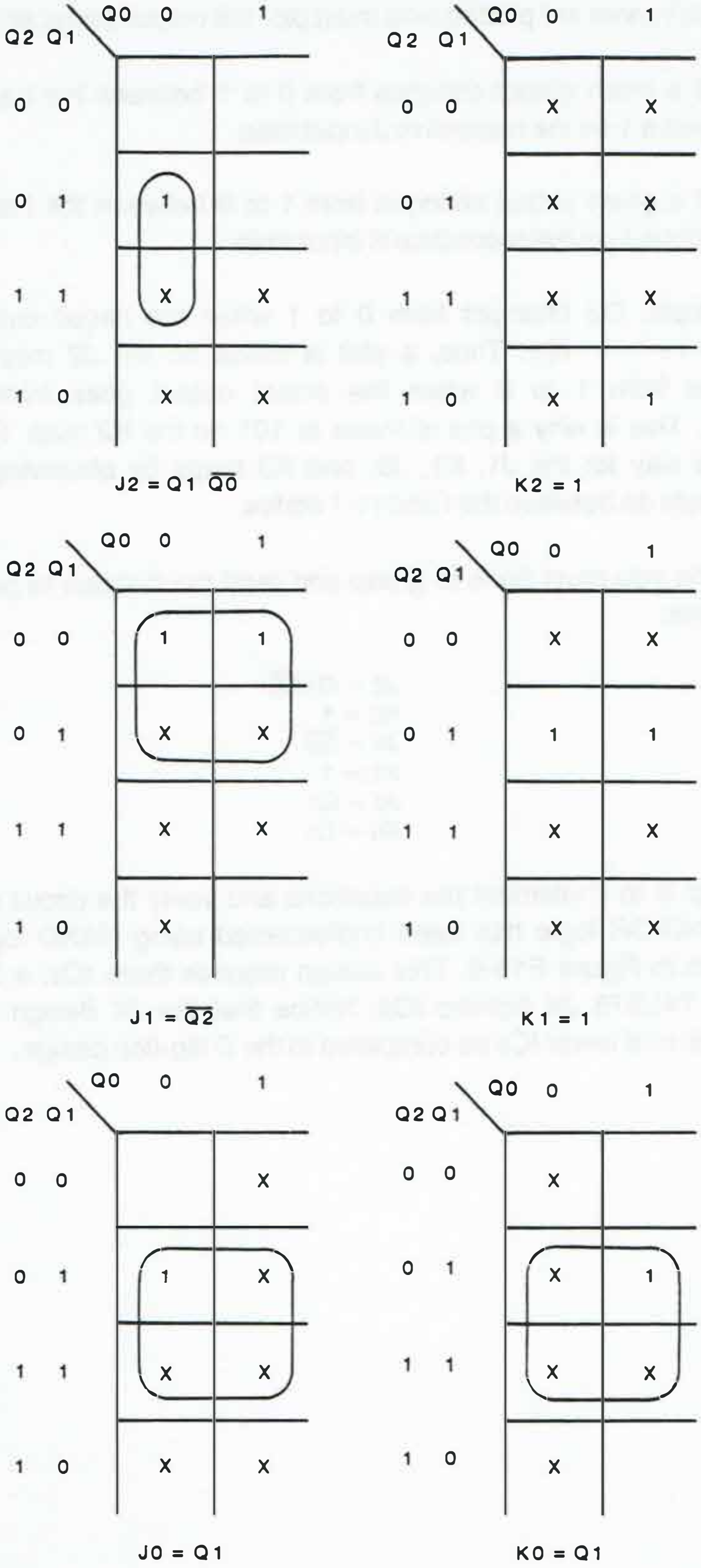
4. Switch power OFF and remove the earlier circuit components and wires. Then construct your JK design. Switch power ON and verify that it generates the required sequence.

Discussion

The problem definition and state table steps of the design process for the JK design are the same as those for the D flip-flop design. The difference is that the feedback logic must be altered for the JK design. The first thing you must do for the JK design is to construct a K-map for each of the J and K inputs as shown in Figure E13-5. The don't care conditions from the state table are plotted as X's on all the grids. Then, each grid has additional don't care plots per Table E13-1. This is a repeat of Table 4-4 in Unit 4. Recall that the J2 don't cares occur when Q2 is high and the K2 don't cares occur when Q2 is low. Likewise the J1 don't cares occur when Q1 is high and the K1 don't cares occur when Q1 is low. The J0 don't cares are when Q0 is high, while the K0 don't cares occur when Q0 is low.

Table E13-1: J and K Don't Care Conditions

	Q2	Q1	Q0		Q2	Q1	Q0
J2 Don't Cares:	1	0	0	K2 Don't Cares:	0	0	0
	1	0	1		0	0	1
	1	1	0		0	1	0
	1	1	1		0	1	1
J1 Don't Cares:	0	1	0	K1 Don't Cares:	0	0	0
	0	1	1		0	0	1
	1	1	0		1	0	0
	1	1	1		1	0	1
J0 Don't Cares:	0	0	1	K0 Don't Cares:	0	0	0
	0	1	1		0	1	0
	1	0	1		1	0	0
	1	1	1		1	1	0



Q2 Q1 \ Q0	0	1
0 0		
0 1	1	
1 1	X	X
1 0	X	X

$J2 = Q1\,\overline{Q0}$

Q2 Q1 \ Q0	0	1
0 0	X	X
0 1	X	X
1 1	X	X
1 0	X	1

K2 = 1

Q2 Q1 \ Q0	0	1
0 0	1	1
0 1	X	X
1 1	X	X
1 0	X	

$J1 = \overline{Q2}$

Q2 Q1 \ Q0	0	1
0 0	X	X
0 1	1	1
1 1	X	X
1 0	X	

K1 = 1

Q2 Q1 \ Q0	0	1
0 0		X
0 1	1	X
1 1	X	X
1 0	X	X

J0 = Q1

Q2 Q1 \ Q0	0	1
0 0	X	
0 1	X	1
1 1	X	X
1 0	X	

K0 = Q1

Figure E13-5

K-maps for the J and K flip-flop inputs.

Once the don't cares are plotted, you must plot the output states as follows:

1. If a given output changes from 0 to 1 between the t and t+1 states, plot a 1 on the respective J input map.
2. If a given output changes from 1 to 0 between the t and t+1 states, plot a 1 on the respective K input map.

As an example, Q2 changes from 0 to 1 when the circuit output goes from t = 010 to t+1 = 101. Thus, a plot is made on the J2 map at 010. Also, Q2 changes from 1 to 0 when the circuit output goes from t = 101 to t+1 = 001. This is why a plot is made at 101 on the K2 map. Plots are made in the same way for the J1, K1, J0, and K0 maps by observing what the Q1 and Q0 outputs do between the t and t+1 states.

The next thing you must do is to group and read the K-maps to get the following logic equations:

$$
\begin{aligned}
J2 &= Q1\,\overline{Q0} \\
K2 &= 1 \\
J1 &= \overline{Q2} \\
K1 &= 1 \\
J0 &= Q1 \\
K0 &= Q1
\end{aligned}
$$

The final step is to implement the equations and verify the circuit operation. The preceding AND/OR logic has been implemented using NAND logic in the final design shown in Figure E13-6. This design requires three ICs: a 74LS00 NAND IC, and two 74LS76 JK flip-flop ICs. Notice that the JK design results in less feedback logic and fewer ICs as compared to the D flip-flop design.

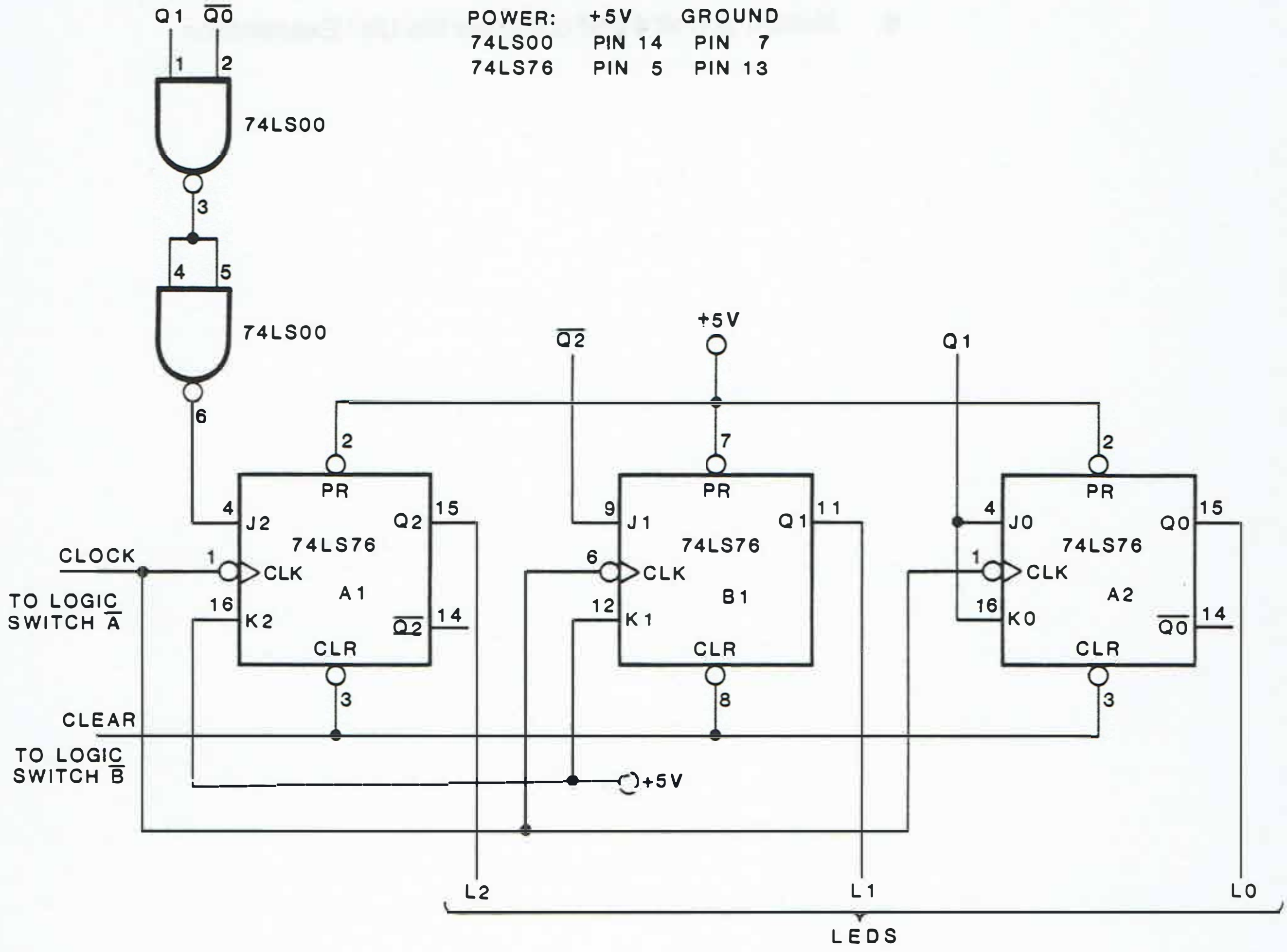


Figure E13-6

The final JK design.

Procedure (continued)

5. Switch power OFF and remove the circuit components and wire.

6. Return to Unit 4 and complete the Unit Examination.

EXPERIMENT 14

DISPLAY CIRCUITS

Objectives:

To demonstrate the operation of a 7-segment LED display circuit.

To show how a decoder/driver is used to convert BCD code to 7-segment code.

To construct a multi-digit display by cascading two single-digit display circuits.

To demonstrate the ripple blanking feature of a display circuit.

Introduction

One of the simplest ways to display digital data is to use a 7-segment display. The display consists of seven independent display segments arranged in a figure eight (8) fashion. These seven segments can be combined to display the decimal digits 0 through 9, as well as many alphabetic characters.

Seven-segment displays require that binary signals be decoded into a 7-segment character code to provide the logic and signal power required to illuminate the proper display segments. This is the job of the **decoder/driver**.

Individual display stages can be cascaded to produce multi-digit or multi-character displays. In the case of a multi-digit counting display, the most significant output of one display stage must be used to clock the next, and so on. Many times, it is desirable to blank-out leading zeros in a multi-digit display. This is called **ripple blanking** and is usually a feature of the decoder/driver devices used in the display circuit.

In this experiment, you will first construct and demonstrate the operation of a single-digit display circuit. Then, you will cascade two single-digit display circuits to form a multi-digit display.

Materials Required

Heathkit Engineering Design Trainer ET-1000
2 — 7447 Seven-Segment Decoder Driver
2 — 74LS90 Decade Counter
2 — 5082-7731 Common Cathode 7-Segment LED
2 — 220 Ohm 1/4-Watt Resistor(Red-Red-Brown-Gold)

Procedure

1. Construct the single-stage 7-segment display circuit shown in Figure E14-1. However, DO NOT connect the generator output to pin 14 of the 74LS90 counter at this time. Be sure to use the upper breadboard block for this circuit. You will be using the lower block later in the experiment.

   As you construct the circuit, note that there is a 220 ohm current limiting resistor connected between +5-volts and the anode pins of the LED. Also notice that not all of the pins are physically present on the LED. However, the missing pins are counted for determining the pin numbers, as shown in the LED diagram below the circuit. This is common practice for multi-segment LEDs.

2. Switch power ON and test the LED by pressing then releasing the **A** logic switch. All seven segments of the LED should illuminate while the switch is pressed, if your circuit has been wired properly. Check your circuit if all of the LED segments do not illuminate.

3. Switch power OFF and connect the left (positive/negative transition) square wave output from the Function Generator to pin 14 ($\overline{\text{INPUT A}}$) of the 74LS90 counter. Adjust the generator for a 1 Hz signal. Switch power ON and observe the LED. It should be counting from 0 to 9. Why? ______________________

   ____

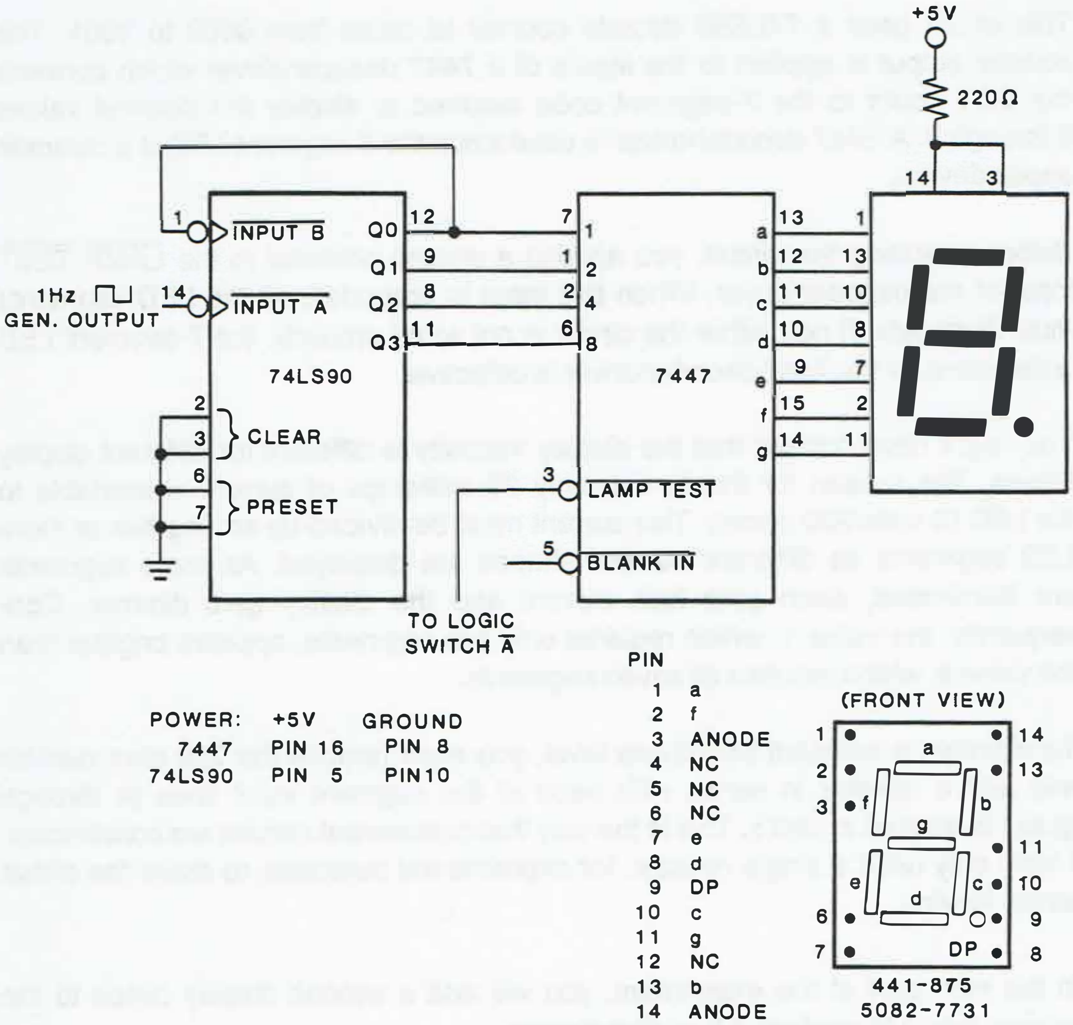


Figure E14-1

A decimal digit display.

Discussion

This circuit uses a 74LS90 decade counter to count from 0000 to 1001. The counter output is applied to the inputs of a 7447 decoder/driver which converts the BCD count to the 7-segment code required to display the decimal values 0 through 9. A 7447 decoder/driver is used since the 7-segment LED is a common anode device.

Before operating the circuit, you applied a ground potential to the $\overline{\text{LAMP TEST}}$ input of the decoder/driver. When this input is grounded, all the LED segments must illuminate. If not, either the circuit is not wired properly, the 7-segment LED is defective, or the 7447 decoder/driver is defective.

You might have noticed that the display intensity is different for different display values. The reason for this is that only 23 milliamps of current is available to the LED (5 volts/220 ohms). This current must be divided-up among two or more LED segments as different number shapes are displayed. As more segments are illuminated, each gets less current and the display gets dimmer. Consequently, the value 1, which requires only two segments, appears brighter than the value 8, which requires all seven segments.

To maintain a constant brightness level, you must remove the 220 ohm resistor and add a resistor in series with each of the segment input lines (a through g) as I illustrated in Unit 5. This is the way that commercial circuits are constructed. I have only used a single resistor, for experimental purposes, to make the circuit easier to wire.

In the next part of the experiment, you will add a second display circuit to the original circuit to produce a two-digit display.

Procedure (continued)

4. Switch power OFF and construct another display circuit identical to the first, on the **lower** breadboard block. As before, do not connect the generator signal to pin 14 of this 74LS90. In addition, connect the LAMP TEST input (pin 3) of this 7447 decoder/driver to the **B** logic switch instead of the **A** logic switch.

5. Switch power ON and test the LED by pressing then releasing the **B** logic switch. All seven segments of the new LED should illuminate while the switch is pressed.

6. Switch power OFF and connect the left (positive/negative transition) square wave output from the Function Generator to pin 14 (INPUT A) of the new 74LS90 counter. Switch power ON and verify that the new circuit works properly.

7. Switch power OFF and remove the wire you just installed between the 1 Hz generator signal and INPUT A of this **second** display circuit. Then connect the two display circuits together with a wire from the Q3 output (pin 11) of the **first** circuit (upper breadboard block) to INPUT A (pin 14) of the **second** circuit (lower breadboard block).

8. Switch power ON and observe the two-digit display. Since the two stages are now combined, or cascaded, you should observe a decimal count from 00 to 99. You might want to turn the FREQ. control up to speed-up the count.

Discussion

This two stage circuit uses two counters to generate a decimal count for each digit of the display. The input clock signal is applied to the least significant counter to generate a display from 0 to 9. Then, when this counter roles over, it clocks the next counter to generate a display from 00 to 99. You might have noticed that leading zero was not blanked-out. For instance, a count of 7 is displayed as 07. How can you provide leading zero blanking for this circuit?

Procedure (continued)

9. Switch power OFF and connect a wire from the $\overline{\text{BLANK IN}}$ line (pin 5) of the most significant display/driver (lower breadboard block) to ground.

10. Switch power ON and observe the display. You should see that the leading zero is now blanked.

Discussion

The $\overline{\text{BLANK IN}}$ control line on the 7447 decoder/driver provides ripple blanking for the display. Tying this line low turns the display off for a BCD input of 0000. As a result, the most significant second stage display never displays the value 0. If more than two stages are constructed, the $\overline{\text{BLANK IN}}$ line of the most significant stage is tied to ground. Then, the $\overline{\text{BLANK OUT}}$ line of the most significant 7447 must be connected to the $\overline{\text{BLANK IN}}$ line of the next 7447, and so on. With this arrangement, any leading zeros are automatically blanked-out of the display. However, any zeros within the value are not blanked.

Procedure (continued)

11. Switch power OFF and remove the circuit components and wire.

12. Return to Unit 5 and complete the Unit Examination.

EXPERIMENT 15

DIGITAL-TO-ANALOG CONVERSION

Objectives:

To construct a 6-bit binary-weighted DAC using a resistor ladder circuit.

To investigate the properties of a current-to-voltage op amp circuit.

To calculate and measure the voltage output of a DAC circuit for various binary input values.

Introduction

As you know, each bit position within a binary number represents a power of 2. The least significant bit position represents 2 raised to the zero power, the next bit position represents 2 raised to the first power, and so on. In other words, a given bit within a binary number is **weighted** by powers of 2 according to its position within the number. Digital-to-analog converters use this idea of positional weighting to convert a digital value to a proportional analog value. A parallel resistor ladder circuit is constructed where the resistor values are weighted in powers of 2. Applying a binary input to the ladder results in currents within the ladder that are weighted by powers of 2. The sum of the weighted currents out of the ladder form an analog signal that is proportional to the digital input value. Such a DAC is called a **binary-weighted DAC.**

In this experiment, you will construct a 6-bit binary weighted resistor ladder DAC circuit using ordinary resistors and a 741C general purpose op amp. You will calculate the expected DAC circuit output for several digital input values. Then, you will verify the actual output of the DAC by measurement.

Materials Required

Heathkit Engineering Design Trainer ET-1000
2 — 10 Kilohm, 1/4-Watt, 5% Resistor (Brown-Black-Orange-Gold)
1 — 20 Kilohm, 1/4-Watt 5% Resistor (Red-Black-Orange-Gold)
1 — 40 Kilohm, 1/4-Watt, 1% Resistor (Yellow-Black-Black-Red-Brown)
1 — 80 Kilohm, 1/4-Watt, 1% Resistor (Gray-Black-Black-Red-Brown)
1 — 162 Kilohm, 1/4-Watt, 1% Resistor (Brown-Blue-Red-Orange-Brown)
1 — 320 Kilohm, 1/4-Watt, 1% Resistor (Orange-Red-Black-Orange-Brown)
1 — 741C Op Amp (443-22)
1 — VOM, VTVM, DVM, or Oscilloscope

Procedure

1. Construct the circuit shown in Figure E15-1. Notice that the op amp is connected to the + 12-volt and − 12-volt power supply lines as well as ground. Do not connect the six resistors to the + 5-volt supply line at this time.

   This is a 6-bit resistive ladder DAC circuit. The 320 k~ resistor, at position 0, controls the least significant bit, or LSB, input to the DAC. The 10 k~ resistor, at position 5, controls the most significant bit, or MSB, input to the DAC.

   NOTE: When I tell you to measure a voltage with a VOM, in the following steps, you may use a VOM, VTVM, DVM, or oscilloscope.

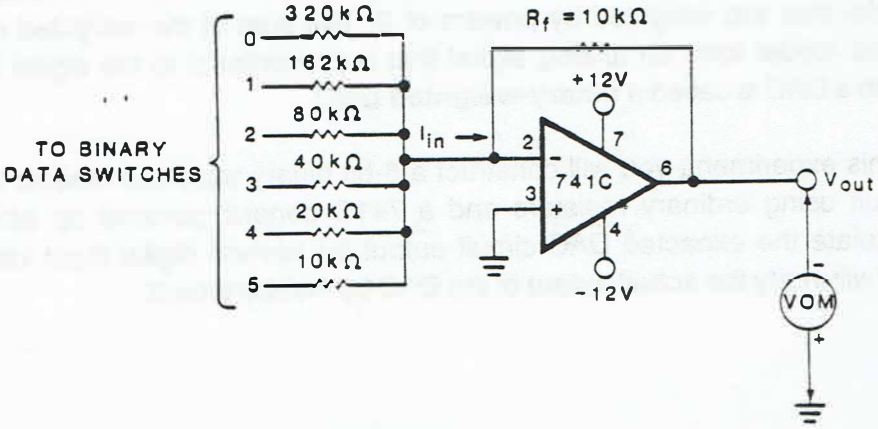


Figure E15-1

A 6-bit binary weighted register ladder DAC circuit.

2. Connect the negative (−) lead of a VOM to the output line of the 741C op amp (pin 6), and connect the positive (+) lead of the VOM to ground. Set the VOM to a +DC scale that will permit you to measure a +5-volt level.

3. Calculate the current passing through each individual ladder resistor when it is connected to +5 volts. Enter the calculated current value in the first empty column of Table E15-1.

   For example, you will be applying +5 volts to each ladder resistor to produce a logic 1 response at that input. Consequently, by Ohm's law, binary bit 5 will produce 5 volts/10 kilohms, or 0.5 milliamps, of current through its ladder resistor.

Table E15-1: Calculated and measured output contributions for the individual input bits of the DAC circuit in Figure E15-1.

Binary Bit Number	Calculated Resistor Ladder Branch Current (I)	Calculated Output Voltage Contribution (I × 10 k)	Measured Output Voltage Contribution
5	________	________	________
4	________	________	________
3	________	________	________
2	________	________	________
1	________	________	________
0	________	________	________

   You should find that the currents are weighted by a factor of 2. Each successive current should be one-half that of the preceding current, since its ladder resistor is twice that of the preceding resistor.

4. Now, calculate the output voltage contributed by each binary logic bit by multiplying the respective ladder current by the op amp feedback resistor value of 10 kilohms.

   As an example, the output voltage contribution of binary bit 5 must be 0.5 milliamps × 10 kilohms, or 5 volts. Because of the binary weighting of the resistor ladder, each successive output voltage contribution must be one-half that of the previous one.

   Enter the calculated values in the second column of Table E15-1.

5. Switch power ON and apply +5 volts to the 10 k~ resistor at bit position 5. Measure and record the voltage output obtained from the op amp in the last column of Table E15-1. The measured value should be close to your calculated value in Table E15-1.

6. Remove power from the 10 k~ resistor at bit position 5.

7. Apply +5 volts to the 20 k~ resistor at bit position 4. Measure and record the voltage output obtained from the op amp for binary bit 4.

8. Repeat the preceding measurement procedure for each of the remaining binary data bit resistors. For accurate results, you should set your VOM to the smallest voltage range that allows you to measure the given output voltage.

Discussion

The six resistors in the DAC circuit form a 6-bit binary weighted resistor ladder. The most significant bit, or MSB, resistance is the lowest resistor value (10 kilohms) and, thus, produces the largest branch current. Each successive branch resistance is twice the value of the preceding branch resistance, thereby weighting the currents in powers of two. The least significant bit, or LSB, resistance is the highest resistor value (320 kilohms) and produces the smallest branch current.

The op amp circuit converts the output current of the ladder to a voltage output according to Ohm's law. Given a current output from the ladder circuit, the corresponding voltage output is found by multiplying the given current value by the op amp feedback resistor value. The voltage obtained from the op amp is negative since the circuit inverts the input current value. As a result, $V_{out} = -I_{IN}R_f$ for this circuit.

You might have noticed some discrepancy between your calculated and measured values in Table E15-1. Why? Most of the error can be contributed to circuit loading, meter loading, and resistor tolerance. For these and other reasons, binary weighted resistor ladder DAC circuits are not as common in industry as other types of DAC circuits.

You may have also wondered why you didn't use the binary data switches on the Trainer to control the voltage going to each ladder resistor. If you have a schematic of the Trainer, you can see why. Each switch is connected to +5 volts through a 4700 ohm series resistor. That much resistance added to each ladder resistor would have had a significant affect on the output of your DAC, especially at the MSB end of the ladder circuit.

Procedure (continued)

9. Calculate the analog output voltage obtained from the DAC circuit for each of the digital input values listed in Table E15-2. Note that a logic 1 represents +5 volts applied to a ladder resistor and a logic 0 represents no voltage. You don't have to connect a ladder resistor to ground to have a logic 0.

Table E15-2: Calculated versus measured output data for different digital input values.

Digital Input Value	Calculated Analog Voltage Output Value	Measured Analog Voltage Output Value
00 0110	________	________
00 1010	________	________
00 1100	________	________
01 0100	________	________
01 0111	________	________
01 1111	________	________
10 1000	________	________
10 1001	________	________
11 0000	________	________
11 1111	________	________

10. Use the output of the +5-volt supply to produce each of the digital input values listed in Table E15-2. Measure and record the analog output voltage obtained from the DAC circuit. Compare your calculated and measured values.

Discussion

Given a binary input value to the DAC circuit, you can find the proportional output voltage level obtained from the DAC by summing the individual input bit contributions from Table E15-1. Your measured values in Table E15-2 should compare very closely to the calculated values if you used the individual bit voltage values that were measured and recorded in Table E15-1.

Procedure (continued)

11. Switch power OFF and remove the circuit components and wire.
12. Proceed to Experiment 16.

EXPERIMENT 16

ANALOG-TO-DIGITAL CONVERSION

OBJECTIVES:

To introduce monolithic ADC devices and demonstrate the operation of the ADC0804 ADC and 9402 V/F converter.

To show how to convert a 0 to 5-volt analog signal to an 8-bit binary value using an ADC.

To demonstrate the use of the START and EOC control lines of an ADC.

To show how a look-up table is used to interpret digital output values from an ADC.

To graph the transfer function of a V/F converter.

Introduction

An analog-to-digital converter is a device that converts a continuous analog voltage signal to a 2-state digital signal that can be processed by a digital circuit. The digital output of the converter is always proportional to the analog input value.

In general, there are two classes of analog-to-digital converters: the parallel output ADC and serial output V/F converter. A parallel output ADC converts an analog input signal to a multi-bit output word whose value is proportional to the analog input. A serial output V/F converter generates a train of serial output pulses whose frequency is proportional to the analog input.

In this experiment, you will investigate the operation of both a parallel output ADC and a serial output V/F converter. You will calculate the expected converter output for a given analog input, then compare this value to the actual measured output.

Materials Required

Heathkit Engineering Design Trainer ET-1000
1 — ADC0804 Analog-to-Digital Converter
1 — 9402 V/F Converter
1 — 1N5231B 5.1-Volt Zener Diode
1 — 150 pF Capacitor
1 — 220 pF Capacitor
1 — 1000 pF (0.001 μF) Capacitor
1 — 1000 Ohm, 1/4-Watt, 5% Resistor (Brown-Black-Red-Gold)
2 — 10 Kilohm, 1/4-Watt, 5% Resistor (Brown-Black-Orange-Gold)
1 — 100 Kilohm, 1/4-Watt, 5% Resistor (Brown-Black-Yellow-Gold)
1 — 500 Kilohm, 1/4-Watt, 1% Resistor (Green-Black-Black-Orange-Brown)
1 — 10 Kilohm Potentiometer
1 — VOM, VTVM, or DVM
1 — Oscilloscope

Procedure

1. Construct the circuit shown in Figure E16-1. Use **caution** when handling the ADC0804, since it is a MOS device and sensitive to static electricity. Make sure you are grounded, and avoid touching the pins.

2. Switch power ON and observe the LED display. You should observe a random display pattern on the LEDs.

3. Turn the 10 kilohm potentiometer control (wiper) in either direction and observe the LED display. Turning the wiper in one direction will increase the digital output display value, while turning in the opposite direction will decrease the digital display value.

4. Set your voltmeter to a range that will allow you to measure a +5-volt level.

5. Connect your voltmeter between the 10 kilohm potentiometer wiper (pin 6 on the ADC) and ground.

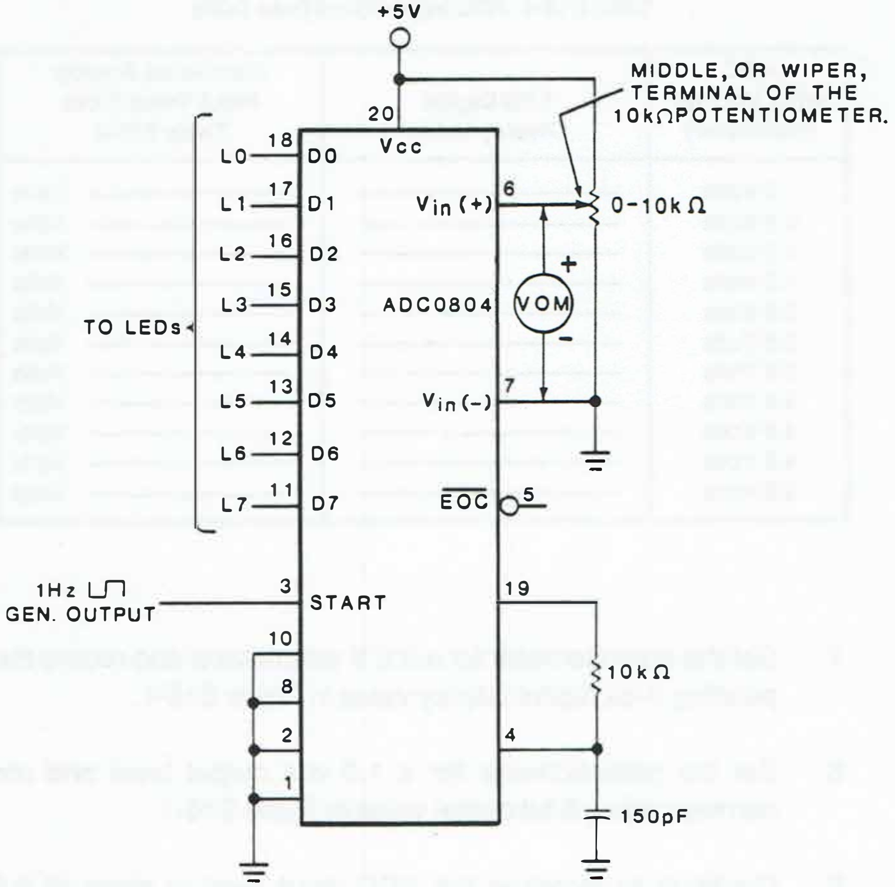


Figure E16-1

An 8-bit parallel output ADC.

6. Set the potentiometer for a 0 V output and record the corresponding 8-bit digital display value in Table E16-1. NOTE: you should observe a digital display value of 0000 0000, as indicated by the ADC output LEDs.

Table E16-1: ADC Input/Output Data Table

ADC Input Voltage (Voltmeter)	LED Digital Display Value	Calculated Analog Input Value From Table E16-2
0 Volts	________________	____________ Volts
0.5 Volts	________________	____________ Volts
1.0 Volts	________________	____________ Volts
1.5 Volts	________________	____________ Volts
2.0 Volts	________________	____________ Volts
2.5 Volts	________________	____________ Volts
3.0 Volts	________________	____________ Volts
3.5 Volts	________________	____________ Volts
4.0 Volts	________________	____________ Volts
4.5 Volts	________________	____________ Volts
5.0 Volts	________________	____________ Volts

7. Set the potentiometer for a 0.5 V output level and record the corresponding 8-bit digital display value in Table E16-1.

8. Set the potentiometer for a 1.0 volt output level and record the corresponding 8-bit digital value in Table E16-1.

9. Continue to increase the ADC input level in steps of 0.5 V and record the corresponding 8-bit ADC digital output value in Table E16-1.

10. Using Table E16-2 and your digital display values from Table E16-1, calculate the expected analog input level. Record each of the calculated voltage values in the third column of Table E16-1 next to the respective digital display value.

Table E16-2: Decoding Table for the Digital Output LEDs

Binary	Hex	Most Significant Voltage Value (V_{MS})	Least Significant Voltage Value (V_{LS})
0000	0	0	0
0001	1	0.32	0.02
0010	2	0.64	0.04
0011	3	0.96	0.06
0100	4	1.28	0.08
0101	5	1.60	0.10
0110	6	1.92	0.12
0111	7	2.24	0.14
1000	8	2.56	0.16
1001	9	2.88	0.18
1010	A	3.20	0.20
1011	B	3.52	0.22
1100	C	3.84	0.24
1101	D	4.16	0.26
1110	E	4.48	0.28
1111	F	4.80	0.30

For example, suppose you have a digital value of 0100 1011. To decode this value using Table E16-2, you must obtain a most significant voltage value (V_{MS}) using the four most significant digital bits, and a least significant voltage value (V_{LS}) using the four least significant digital bits. The final voltage value (V) is then the sum of the most significant voltage value (V_{MS}) and the least significant voltage value (V_{LS}), or:

$$V = V_{MS} + V_{LS}$$

Thus, if the LED display indicates a digital value of 0100 1011: Table E16-2 shows that $V_{MS} = 1.28$ volts, which corresponds to 0100; and $V_{LS} = 0.22$ volts, which corresponds to 1011. Thus, the final voltage value is:

$$\begin{aligned} V &= V_{MS} + V_{LS} \\ &= 1.28 \text{ volts} + 0.22 \text{ volts} \\ &= 1.5 \text{ volts} \end{aligned}$$

11. Compare your calculated values (column 3 of Table E16-1) to your measured values (column 1 of Table E16-1). You should find that the two values compare rather closely. Any differences can be attributed to calculating error, measurement error, and meter loading effects.

Discussion

In the first part of this experiment, you have demonstrated the general operation of an ADC by converting a 0 to 5-volt analog input signal to a proportional 8-bit digital output value. The maximum input voltage level for this particular ADC is 5 volts. Therefore, as shown in Table E16-2, the least significant digital output bit has a resolution of 1/256 times 5 volts, or 0.02 volts. Table 16-2 was determined by calculating and adding the individual bit weights in this manner. As a result, you can use Table E16-2 to interpret any digital output value from the ADC.

Table E16-2 is called a **look-up table** since, given a digital output value from the ADC, you must "look-up" its corresponding analog value to determine the input voltage level to the ADC. When you interface an ADC to a microprocessor, this look-up table must be placed in memory so that the microprocessor can interpret the digital output value from the ADC. For obvious reasons, it is then called a **memory** look-up table.

A closer look at the circuit diagram in Figure E16-1 reveals that you are using the 1 Hz clock signal from the Trainer to **start** the conversion. Consequently, the digital output display is up-dated once every second.

Procedure (continued)

12. Set the FREQ. MULTIPLIER switch to 100.

13. Randomly change the potentiometer setting and observe the output LEDs. You should observe a quicker response to the potentiometer changes than before, since the ADC is now making 100 conversions per second.

14. Switch power OFF. Remove the wire connecting the START line (pin3) of the ADC to the square wave output of the Function Generator section of the Trainer. Then connect a wire between the START line (pin 3) and the $\overline{\text{EOC}}$ line (pin 5) of the ADC.

15. Switch power ON and randomly change the potentiometer setting. Observe the output LEDs and you should find that the digital output responds very quickly to changes in the analog input level.

Discussion

Recall that the ADC START line controls the conversion process. Each time a low-to-high transition is applied to the START line, the ADC will make a new conversion. By changing the START line connection to 100 Hz in step 12, the ADC was forced to make one hundred conversions per second, thereby providing a faster conversion response time. Prior to this, the START line was connected to a 1 Hz signal.

In step 14, you jumpered the ADC START line to the end of conversion, or $\overline{\text{EOC}}$, line. This connection produces the maximum number of conversions per second, since each time a conversion is completed the $\overline{\text{EOC}}$ line generates a signal to the START line to begin a new conversion. With this connection, the ADC is making conversions at approximately an 8.6 kHz rate, or 8600 conversions per second.

Procedure (continued)

16. Switch power OFF, remove the ADC circuit, and construct the V/F converter circuit shown in Figure E16-2. Notice that the −5-volt input to pin 4 of the V/F comes from the reference voltage supply at pin 7.

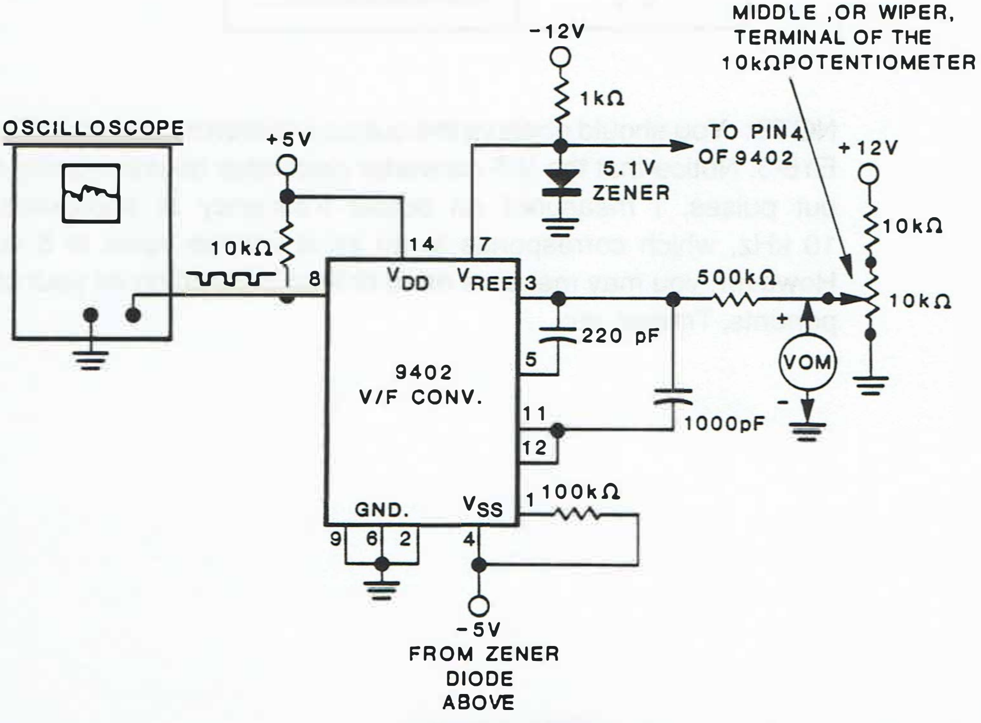


Figure E16-2

A V/F converter circuit.

17. Connect your voltmeter between the wiper, or middle, terminal of the 10 kilohm potentiometer and ground as shown. Set the voltmeter to a range that will permit you to measure +6 volts.

18. Connect your oscilloscope between pin 8 of the V/F converter and ground as shown. Set the volts/division scale to 1 V/division and the time base to 20 μS/division.

19. Switch power ON and adjust the potentiometer to provide a +6-volt input to the V/F converter.

20. Observe the V/F converter output on the oscilloscope display. Measure and record the output frequency value in Table E16-3 across from the 5 V input voltage value in the table.

Table E16-3: V/F Converter Input/Output Data Table

V/F Converter Input Voltage (voltmeter)	**V/F Converter Output Frequency (oscilloscope)**
5 V	________
4 V	________
3 V	________
2 V	________
1 V	________
0 V	________

NOTE: You should observe the output waveform pictured in Figure E16-3. Notice that the V/F converter generates negative-going output pulses. I measured an output frequency of approximately 10 kHz, which corresponds to an input voltage value of 5 volts. However, you may measure more or less, depending on your components, Trainer, etc.

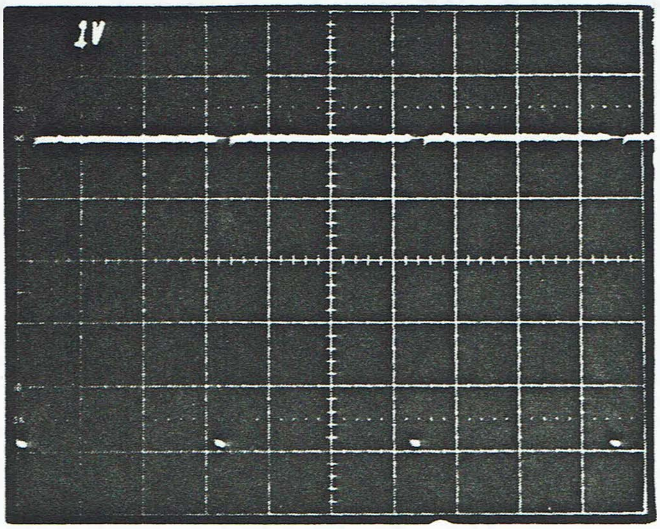


Figure E16-3

Oscilloscope display of the V/F converter output.

21. Set the potentiometer to provide a +4-volt input to the V/F converter.

22. Measure and record the V/F converter output frequency in Table E16-3 across from the 4 V input voltage value.

23. Set the potentiometer to provide a +3-volt input to the V/F converter.

24. Measure and record the V/F converter output frequency in Table E16-3.

25. Continue to decrease the V/F converter input in steps of 1 volt and record the corresponding frequency output value in Table E16-3. Notice that, with a 1 volt input level, the output frequency of the V/F converter drops to approximately 1.6 kHz. With a 0 volt input level, the V/F converter output frequency is 0 Hz.

26. Now, plot the V/F converter input/output values from Table E16-3 on the graph provided in Figure E16-4. Connect the plotted points and you should find a straight line graph, or transfer function.

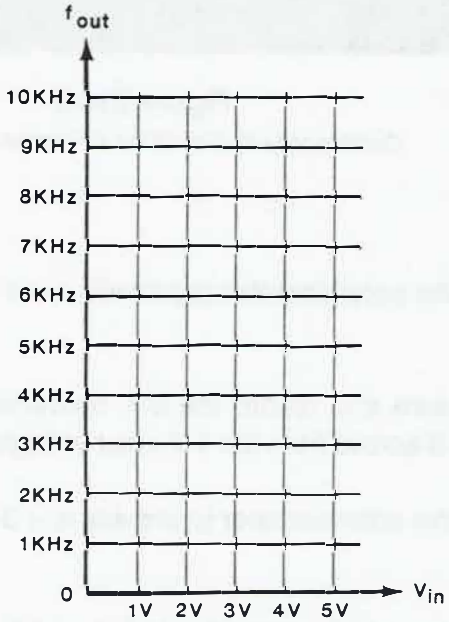


Figure E16-4

Graph to plot V/F input/output values from Table E16-3.

Discussion

The 9402 V/F converter is typical of many relatively low cost V/F converters available on the market today. As you discovered in the above experiment, the 9402 generates a series of negative-going output pulses whose frequency is proportional to the input voltage level. You found that the 9402 generated frequencies from 0 to approximately 10 kHz corresponding to an input voltage range of 0 to 5 volts. In fact, from the data you plotted in Figure E16-4, you found that the 9402 produced a very linear transfer function. This characteristic of V/F converters makes them desirable over parallel-output ADCs for precise measurement and instrumentation applications.

Procedure (continued)

27. Switch power OFF and remove the circuit components and wire.

28. Return to Unit 6 and complete the Unit Examination.

APPENDIX A

BINARY NUMBERS AND COMPUTER ARITHMETIC

CONTENTS

Introduction . . . . Page A-3

Numbers and Codes . . . . Page A-3

Digital Computer Arithmetic . . . . Page A-19

Self-Test Review . . . . Page A-26

Answers . . . . Page A-30

INTRODUCTION

Digital computers and the digital circuits they are made from are nothing more than electronic switching circuits. These electronic switches are actually integrated transistors that are made to operate in one of two states: conducting or not conducting. In other words, a given integrated transistor is either on or off at any given time. Decisions and computations are made by combining all of these on/off states into a **logical network.**

Since any digital circuit can be reduced to a logical network made up of two-state on/off elements, the **binary** number system is ideal for describing the circuit operation. The **bi** in binary means "two". Thus, like the digital computer, the binary number system is a two-state system. The two states are 1 and 0. Relating the binary system to the operation of a digital circuit, a binary 1 can be used to represent the on state, while a binary 0 can be used to represent the off state. As a result, the operation of any digital circuit can be represented, or described, using the binary number system.

This Appendix has been written to provide you with an introduction to the binary number system and computer arithmetic. You must be familiar with this material before beginning the next portion of this course. I suggest that you read this material and work the problems at the end of the Appendix before going to Unit 1. Even if you have already been exposed to the binary number system, a good review won't hurt.

NUMBERS AND CODES

Let's take a trip back in time to when you were learning how to count. If you learned like I did, you probably began by counting on your fingers. Then, when you ran out of fingers, you used your toes. Counting to 20 posed no problem using this method. But, what about counting to 21? Counting above 20 creates a real dilemma, since you soon run out of fingers and toes.

When using your fingers and toes to count, each finger and toe must represent one given count value, or number. As a result, you would have to have an infinite number of fingers and toes to represent all possible numbers. Now suppose, rather than using fingers and toes to represent numbers, you create several written symbols that can be used to represent all possible numbers. That is, a given symbol represents a given number where each number symbol is unique. The problem with such a system is, like counting with fingers and toes, you soon run out of number symbols. An infinite number of symbols would be required to represent all possible numbers.

Decimal Numbers

Most likely your first or second grade teacher quickly relieved your dilemma by introducing the **positional weight** decimal number system. You found that the decimal number system consists of only ten (dec) number symbols, 0 through 9. You learned that this system starts with a **decimal point**, with each position to the left and right of the decimal point **weighted** according to multiples of 10 like this:

$$\ldots\ 1000\ 100\ 10\ 1\ \underset{\substack{\uparrow \\ \text{DECIMAL POINT}}}{.}\ 1/10\ 1/100\ 1/1000\ \ldots$$

Here, the first position to the left of the decimal point is the 1's position, the next position is the 10's position, the next the 100's position, and so on. The first position to the right of the decimal point is the one-tenth's position, the next position is the one-hundredth's position, the next the one-thousandth's position, and so forth. Using this idea, you found that a decimal number such as 537 consists of 7 ones, 3 tens, and 5 hundreds. To translate the number, you would add $(7 \times 1) + (3 \times 10) + (5 \times 100)$ to get 537, right?

Later on in your education, probably in a high school algebra class, you learned that each position in a decimal number represents a power of ten like this:

$$\ldots\ 10^3\ 10^2\ 10^1\ 10^0\ \underset{\substack{\uparrow \\ \text{Decimal Point}}}{.}\ 10^{-1}\ 10^{-2}\ 10^{-3}\ \ldots$$

The first position to the left of the decimal point is the 1's, or 10^0 position, the next position is the 10's, or 10^1, position, the next is the 100's, or 10^2 position, and so on. The positions to the right of the decimal point are equivalent to negative powers of 10. Using this idea, the number 537 can be translated as:

$$\begin{aligned} 537 &= (7 \times 10^0) + (3 \times 10^1) + (5 \times 10^2) \\ &= (7 \times 1) + (3 \times 10) + (5 \times 100) \\ &= 7 + 30 + 500 \\ &= 537 \end{aligned}$$

The important thing to remember here is that the decimal number system is a positional weight system that can be reduced to powers of 10.

Binary Numbers

You are probably wondering why I went through the above discussion when you have been working with decimal numbers most of your life. The reason is that all positional weight number systems operate the same way. The only difference is the **base** value of the system. For decimal numbers, the base value is 10. For binary numbers, the base value is 2. Consequently, since binary is a positional weight system, any binary number can be reduced to powers of 2 like this:

$$\ldots\ 2^3\ 2^2\ 2^1\ 2^0\ .\ 2^{-1}\ 2^{-2}\ 2^{-3}\ \ldots$$

↑
Binary Point

Here, the powers of 10 used for the decimal system have been replaced with powers of 2 in the binary system. The first position to the left of the **binary point** is the 2^0, or 1's position, the next position is the 2^1, or 2's, position, the next is the 2^2, or 4's, position, followed by the 2^3, or 8's, position, and so on. Thus, each successive position to the left of the binary point is two times the weight of the previous position. Each successive position to the right of the binary point is one-half that of the previous position.

Since the "bi" in binary means two, there are only two symbols used in binary numbers, 1 and 0. To translate a binary number, you use the above powers of 2 the same as you used powers of 10 for decimal translations. For example, the binary number 1101 is translated as follows:

$$\begin{aligned}1101 &= (1 \times 2^0) + (0 \times 2^1) + (1 \times 2^2) + (1 \times 2^3)\\ &= (1 \times 1) + (0 \times 2) + (1 \times 4) + (1 \times 8)\\ &= 1 + 0 + 4 + 8\\ &= 13\end{aligned}$$

Example A-1

Convert the following binary numbers to decimal.

a. 1001
b. 0111
c. 1111 1111
d. 1100 0101

Using powers of 2 according to the positions of the 1's and 0's in the number, you get:

$$\begin{aligned}\text{a. } 1001 &= (1 \times 2^0) + (0 \times 2^1) + (0 \times 2^2) + (1 \times 2^3)\\ &= 1 + 0 + 0 + 8\\ &= 9\end{aligned}$$

b. $0111 = (1 \times 2^0) + (1 \times 2^1) + (1 \times 2^2) + (0 \times 2^3)$
$= 1 + 2 + 4 + 0$
$= 7$

To convert the next two numbers, I will drop the power of 2 notation, remembering that each successive position in the number is twice that of the previous position. Using this idea,

c. $1111\,1111 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128$
$= 255$

d. $1100\,0101 = 1 + 0 + 4 + 0 + 0 + 0 + 64 + 128$
$= 197$

In this example, you started with a binary number and converted it to a decimal equivalent. Now, let's go the other way and see how decimal numbers are converted to binary numbers. The method used for decimal-to-binary conversions is called the **double-dabble** method. The basic idea is to start with the decimal number, divide it by two and record the remainder. The quotient generated by the first division is then divided by two and the remainder is recorded again. This successive division by two process is continued until a quotient of 0 is obtained.

For example, suppose you must convert the decimal number 14 to its binary equivalent. The decimal value 14 is first divided by 2 to get 7 with remainder 0. Then, 7 is divided by 2 to get 3 with remainder 1. Next, 3 is divided by 2 to get 1 with remainder 1. Finally, 1 is divided by 2 to get 0 with remainder 1. This successive double-dabble division process is summarized below:

14/2 = 7 with remainder 0 ↑
7/2 = 3 with remainder 1
3/2 = 1 with remainder 1
1/2 = 0 with remainder 1
READ UP

Notice that the division process continues until a quotient of 0 is obtained. The equivalent binary value is then found by reading up in the tabulation, with the most significant digit first. In other words, the decimal value 14 is equivalent to the binary value 1110.

Example A-2

Convert the following decimal numbers to their binary equivalents.

a. 12
b. 35
c. 150

Using the above double-dabble method you get:

a. 12/2 = 6 with remainder 0 ↑
6/2 = 3 with remainder 0
3/2 = 1 with remainder 1
1/2 = 0 with remainder 1
READ UP

Thus, $12_{10} = 1100_2$. Notice that a subscript of 10 is used to denote the decimal number and a subscript of 2 is used to denote the binary number. This will be done from now on, when we need to distinguish between different base numbers.

b. 35/2 = 17 with remainder 1 ↑
17/2 = 8 with remainder 1
8/2 = 4 with remainder 0
4/2 = 2 with remainder 0
2/2 = 1 with remainder 0
1/2 = 0 with remainder 1
READ UP

Thus, $35_{10} = 10\,0011_2$.

c. 150/2 = 75 with remainder 0 ↑
75/2 = 37 with remainder 1
37/2 = 18 with remainder 1
18/2 = 9 with remainder 0
9/2 = 4 with remainder 1
4/2 = 2 with remainder 0
2/2 = 1 with remainder 0
1/2 = 0 with remainder 1
READ UP

Thus, $150_{10} = 1001\,0110_2$.

Bits, Bytes, Nibbles, And Words

The terms bits, bytes, nibbles, and words are commonly used to describe the size of binary values. A **bit** is a 1 or a 0. Thus, the binary value 1001 is a 4-bit value. Likewise, the value 1010 1111 is an 8-bit value. The term **byte** is used to describe an 8-bit value. Thus, one byte is 8 bits, two bytes is 16 bits, and so on. A **nibble** is one-half a byte, or 4-bits. Consequently, one byte is two nibbles and two bytes is equivalent to four nibbles.

The definition of a **word** is not as precise as bits, bytes, and nibbles. A word can be any number of bits, depending on how it is associated with a given digital system. For example, a single bit is a 1-bit word, a byte is an 8-bit word, and a nibble is a 4-bit word. In digital computer systems, there are different types of words, such as **data words**, and **address words**. A typical microprocessor system might use 8-bit data words and 16-bit address words. The lesson to be learned here is that you must be careful when using the term **word**, since its definition and size depends on how it is associated with the digital system.

Before leaving binary numbers, you should be familiar with the symbol **K**. In digital computer electronics, the symbol K is used to denote the decimal value 1024 (2^{10}). As an example of its use, suppose you see that a certain personal computer system is advertised as having "64K bytes" of user memory. This means that the system has 64 × 1024, or 65,536 bytes of user memory. Of course, this is equivalent to 8 × 65,536, or 524,288 bits of memory.

Binary Coded Decimal, or BCD

The binary number system described up to this point is often referred to as **straight binary** code. You should be aware that there are other binary codes that are commonly used in digital electronics. One such code is called **binary coded decimal**, or **BCD**. The BCD code is often used in conjunction with digital displays to display numerical characters.

The BCD code is simple — it consists of groups of four bits that represent the decimal numbers 0 through 9, as shown in Table A-1. Notice that the 4-bit binary values 0000 through 1001 are used to represent the decimal values 0 through 9. What happens to the 4-bit binary values above 1001 (1010 through 1111)? They are not defined, and therefore, **illegal** in the BCD system.

Table A-1: BCD vs. Decimal

BCD	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	Illegal BCD Codes
1011	
1100	
1101	
1110	
1111	

The reason for the BCD code is that it makes conversions between binary and decimal easier. To convert a BCD number to a decimal number, you simply divide the BCD value into groups of four bits, beginning with the four least significant bits. Then convert each group directly into its decimal equivalent. Keep in mind that this method of conversion only works between BCD and decimal. You cannot convert straight binary to decimal in this fashion.

Example A-3

Convert the following BCD values to decimal:

a. 0101
b. 1001 0111
c. 0011 0100 1000
d. 1011

Notice that I have divided the bits into 4-bit groups. This is a common practice when dealing with BCD values.

a. The BCD value 0101 is equivalent to the decimal value 5. Thus,

$$0101_{BCD} = 5_{10}$$

b. There are two groups of four bits in this BCD value, 1001 and 0111. Converting these to decimal, you get: 1001 = 9 and 0111 = 7. Consequently,

$$1001\ 0111_{BCD} = 97_{10}$$

c. The BCD value 0011 0100 1000 consists of the 4-bit binary groups 0011, 0100, and 1000. Converting these to decimal, you get: 0011 = 3, 0100 = 4, and 1000 = 8. Combining these you get,

$$0011\ 0100\ 1000_{BCD} = 348_{10}$$

d. The 4-bit value 1011 is illegal in the BCD system. Therefore, it cannot be converted to decimal.

Conversion from decimal to BCD is just the opposite — you simply express each decimal as a 4-bit BCD value.

Example A-4

Convert the following decimal values to binary coded decimal:

a. 2
b. 35
c. 127
d. 6089

Converting each decimal digit to a 4-bit BCD value, you get:

a. $2_{10} = 0010_{BCD}$

b. $3_{10} = 0011_{BCD}$
$5_{10} = 0101_{BCD}$
Thus,
$35_{10} = 0011\ 0101_{BCD}$

c. $1_{10} = 0001_{BCD}$
$2_{10} = 0010_{BCD}$
$7_{10} = 0111_{BCD}$
Therefore,
$127_{10} = 0001\ 0010\ 0111_{BCD}$

d. $6_{10} = 0110_{BCD}$
$0_{10} = 0000_{BCD}$
$8_{10} = 1000_{BCD}$
$9_{10} = 1001_{BCD}$
Consequently,
$6089_{10} = 0110\ 0000\ 1000\ 1001_{BCD}$

ASCII

The abbreviation ASCII stands for **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. The ASCII code was developed as a communications standard for digital computer equipment. It uses seven bits to represent the characters found on a typical typewriter keyboard. A conversion table for the ASCII code is provided in Figure A-1.

	COLUMN	**0 (3)**	**1 (3)**	**2 (3)**	**3**	**4**	**5**	**6**	**7 (3)**
ROW	**BITS 765 4321**	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
10	1010	LF	SUB	*	:	J	Z	j	z
11	1011	VT	ESC	+	;	K	[	k	{
12	1100	FF	FS	,	<	L	\	l	¦
13	1101	CR	GS	–	=	M	]	m	}
14	1110	SO	RS	.	>	N	⌒ (1)	n	~
15	1111	SI	US	/	?	O	_ (2)	o	DEL

Figure A-1A
Table of 7-bit American Standard Code
for Information Interchange.

NOTES:

(1) Depending on the machine using this code, the symbol may be a circumflex, an up-arrow, or a horizontal parenthetical mark.

(2) Depending on the machine using this code, the symbol may be an underline, a back-arrow, or a heart.

(3) Explanation of special control functions in columns 0, 1, 2, and 7.

NUL	Null
SOH	Start of Heading
STX	Start of Text
ETX	End of Text
EOT	End of Transmission
ENQ	Enquiry
ACK	Acknowledge
BEL	Bell (audible signal)
BS	Backspace
HT	Horizontal Tabulation (punched card skip)
LF	Line Feed
VT	Vertical Tabulation
FF	Form Feed
CR	Carriage Return
SO	Shift Out
SI	Shift In
SP	Space (blank)
DLE	Data Link Escape
DC1	Device Control 1
DC2	Device Control 2
DC3	Device Control 3
DC4	Device Control 4
NAK	Negative Acknowledge
SYN	Synchronous Idle
ETB	End of Transmission Block
CAN	Cancel
EM	End of Medium
SUB	Substitute
ESC	Escape
FS	File Separator
GS	Group Separator
RS	Record Separator
US	Unit Separator
DEL	Delete

Figure A-1B

Continuation of the table of 7-bit American Standard Code for Information Interchange.

To find the ASCII code for a given character, you must first locate the character inside the table. Then read up to get the three most significant code bits and across to get the four least significant code bits. For instance, to find the ASCII code for the lower case "a", you find the letter "a" in the table. Reading up, you get 110 as the three most significant code bits. Reading across to the left, you find that 0001 are the four least significant code bits. Thus, the ASCII code for the letter "a" is 110 0001.

ASCII was developed so that equipment produced by different manufacturers will all communicate using the same digital language. It's just like people speaking the same language. If a given group of people all spoke a different language, nobody would know what the other person was saying. In digital communications, if separate pieces of digital equipment communicated using different binary codes, there would have to be code conversion circuits to translate one code to another. However, if all the equipment is designed to "speak" ASCII, no code conversions are necessary.

Hexadecimal, or HEX, Numbers

Another number system that is often used in digital electronics is hexadecimal. Now that you know about decimal and binary numbers, hexadecimal numbers are easy. The prefix "hex" stands for 16. As a result, the hexadecimal number system has a base of 16 and employs sixteen symbols: 0 through 9, A, B, C, D, E, and F. The number listing in Table A-2 shows the decimal and binary equivalents of these hexadecimal symbols. Notice that the hexadecimal values A through F are equivalent to the decimal values 10 through 15, respectively. In addition, a 4-bit binary value is used to represent a given hex symbol, and vice versa.

Table A-2: Hexadecimal, Decimal, and Binary Equivalents

Hexadecimal	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Since the hexadecimal system is a positional weight system with a base of 16, any hex number can be reduced to powers of 16 as follows:

$$\ldots\ 16^3\ 16^2\ 16^1\ 16^0\ \underset{\substack{\uparrow \\ \text{Hex Point}}}{.}\ 16^{-1}\ 16^{-2}\ 16^{-3}\ \ldots$$

Notice that the first position to the left of the "hex point" is the 16^0, or 1's, position, the next position is the 16^1, or 16's, position, next is the 16^2, or 256, position, and so on. Each successive position to the left is 16 times the weight of the previous position. Each successive positional weight to the right of hex point is obtained by dividing the previous position by 16.

To interpret a hexadecimal value, you simply apply the above powers of 16 the same way you used powers of 10 and 2 for decimal and binary numbers. For example, the hex value 2DF is interpreted like this:

$$\begin{aligned} 2DF_{16} &= (F \times 16^0) + (D \times 16^1) + (2 \times 16^2) \\ &= (F \times 1) + (D \times 16) + (2 \times 256) \\ &= (15 \times 1) + (13 \times 16) + (2 \times 256) \\ &= 15 + 208 + 512 \\ &= 735_{10} \end{aligned}$$

Here, you have to get used to the idea that letters are used to represent quantities. During the above interpretation process the given hex letters must be converted to their equivalent decimal quantities (See Table A-2).

Example A-5

Convert the following hexadecimal values to their decimal equivalents:

a. $5C_{16}$
b. $B5_{16}$
c. ABC_{16}

Using the respective powers of 16 and the definition of the 16 hexadecimal symbols you get:

a. $$\begin{aligned} 5C_{16} &= (C \times 16^0) + (5 \times 16^1) \\ &= (C \times 1) + (5 \times 16) \\ &= (12 \times 1) + (5 \times 16) \\ &= 12 + 80 \\ &= 92_{10} \end{aligned}$$

b. $$\begin{aligned} B5_{16} &= (5 \times 16^0) + (B \times 16^1) \\ &= (5 \times 1) + (B \times 16) \\ &= (5 \times 1) + (11 \times 16) \\ &= 5 + 176 \\ &= 181_{10} \end{aligned}$$

c. $$\begin{aligned} ABC_{16} &= (C \times 16^0) + (B \times 16^1) + (A \times 16^2) \\ &= (C \times 1) + (B \times 16) + (A \times 256) \\ &= (12 \times 1) + (11 \times 16) + (10 \times 256) \\ &= 12 + 176 + 2560 \\ &= 2748_{10} \end{aligned}$$

Notice the power of the hex system. The three digit hex value ABC_{16} can be used in place of the four digit decimal value 2748_{16}. In addition, it becomes even more important to designate values with subscripts when mixing decimal, binary, and hex numbers. For instance, the number 1101 can be used to represent three completely different values, depending on whether it is a decimal, binary, or hex number.

DECIMAL-TO-HEX

To convert decimal numbers to hexadecimal, you apply the double-dabble method like you did when converting decimal to binary, except that you must divide by 16 rather than 2. For example, to convert the decimal number 35 to hex, you must first divide 35 by 16 to get 2 with a remainder of 3. Then you divide 2 by 16 to get 0 with a remainder of 2. Tabulating this process you have:

$$\begin{array}{rl} 35/16 = & 2 \text{ with remainder } 3 \quad \uparrow \\ 2/16 = & 0 \text{ with remainder } 2 \quad | \\ & \qquad \text{READ UP} \end{array}$$

By reading the remainders up, in the preceding tabulation, you conclude that $35_{10} = 23_{16}$.

Example A-6

Represent the following decimal numbers in hexadecimal form:

a. 92_{10}
b. 181_{10}
c. 2748_{10}

Using the double-dabble process of successive division by 16 you get:

a. 92/16 = 5 with remainder 12, or C
5/16 = 0 with remainder 5
↑ READ UP

Thus, $92_{10} = 5C_{16}$.

b. 181/16 = 11 with remainder 5
11/16 = 0 with remainder 11, or B
↑ READ UP

Thus, $181_{10} = B5_{16}$.

c. 2748/16 = 171 with remainder 12, or C
171/16 = 10 with remainder 11, or B
10/16 = 0 with remainder 10, or A
↑ READ UP

Thus, $2748_{10} = ABC_{16}$.

Observe that the decimal remainders in the above example must be translated to their hex equivalents to obtain the proper hex designation.

HEX-TO-BINARY

Although digital circuits operate on binary values, many digital systems require hex inputs and display hex output values. This is especially true of simple microprocessor systems. To design and/or troubleshoot such systems, you must be able to convert between hex and binary. To convert hex to binary, you simply express each hex digit as a 4-bit binary value (See Table A-2). For example, the hex value A7 contains the digits A and 7. The 4-bit binary equivalent of A is 1010, while the 4-bit equivalent of 7 is 0111. Thus the hex value A7 is equivalent to the binary value 1010 0111.

Example A-7

Convert each of the following hex values to binary.

a. $5C_{16}$
b. $B5_{16}$
c. ABC_{16}

Expressing each hex digit as a group of four bits you get:

a. $5_{16} = 0101_2$
$C_{16} = 1100_2$

Thus, $5C_{16} = 0101\ 1100_2$.

b. $B_{16} = 1011_2$
$5_{16} = 0101_2$

Thus, $B5_{16} = 1011\ 0101_2$.

c. $A_{16} = 1010_2$
$B_{16} = 1011_2$
$C_{16} = 1100_2$

Thus, $ABC_{16} = 1010\ 1011\ 1100_2$.

BINARY-TO-HEX

You simply reverse the above process to convert binary to hex. That is, divide the binary value into groups of four bits and express each 4-bit group as a hex digit. For instance, the binary value 10 1111 can be expressed using two 4-bit groups as 0010 1111. The first group of four bits, 0010, is equivalent to the hex digit 2. The second group of four bits, 1111, is equivalent to the hex digit F. As a result, the binary value 10 1111 is equivalent to the hex value 2F.

Example A-8

Convert each of the following binary values to hex.

a. $1001\ 1110_2$
b. $11\ 0101\ 1101_2$
c. $1010\ 0110\ 1111\ 0000_2$

Expressing each 4-bit binary group as a hex digit you get:

a. $1001_2 = 9_{16}$
$1110_2 = E_{16}$

Thus, $1001\ 1110_2 = 9E_{16}$.

b. 11 0101 1101 = 0011 0101 1101
$0011_2 = 3_{16}$
$0101_2 = 5_{16}$
$1101_2 = D_{16}$

Thus, $11\ 0101\ 1101_2 = 35D_{16}$.

c. $1010_2 = A_{16}$
$0110_2 = 6_{16}$
$1111_2 = F_{16}$
$0000_2 = 0_{16}$

Thus, $1010\ 0110\ 1111\ 0000_2 = A6F0_{16}$.

Get used to writing your binary values in 4-bit groups as I have done throughout this Appendix. This makes it easier to convert to hex, should the need arise.

DIGITAL COMPUTER ARITHMETIC

Recall that the basic operation of any digital circuit can be reduced to a two-state binary operation. So, when digital computers must perform arithmetic operations, the digital circuits perform the operations in binary, not decimal as you and I are accustomed to. Consequently, to design digital arithmetic circuits, you must understand how to perform arithmetic operations in binary.

In this course, you only need to know how to add and subtract binary numbers. As you will discover in Experiment 4, multiplication and division are simply special cases of addition and subtraction.

Binary Addition

Adding binary values is similar to adding decimal values. Here are some general rules to go by:

1. List the binary numbers in column form so that the respective bit columns are above each other.
2. Beginning with the least significant bit column, count the number of binary 1's in the column, using the following guide lines:
   A. If the count is even, the sum of the column is 0.
   B. If the count is odd, the sum of the column is 1.
3. Generate a binary 1 **carry** into the next column for **each pair** of binary 1's in the current column.
4. Repeat steps 2 and 3 for each column until all columns and carry's have been added.

The best way to teach you binary addition from here is by example. For instance, suppose that you must add the two 4-bit values 1011 and 1110. First, list the two values in column form like this:

$$\begin{array}{lr} \text{Value \#1:} & 1011 \\ \text{Value \#2:} & \underline{+\ 1110} \end{array}$$

Now, the least significant bit (LSB) column is the right-most column. This column contains only one binary 1. Since one is odd, the sum of the column is 1. Or,

Value #1: 1011
Value #2: + 1110
Sum: 1

Next, notice that there are no binary 1 pairs in this first column. Therefore, no carry's are generated and you move on to the second column. Looking at the second column, you find two binary 1's. Since two is even, the sum of this column is 0. Furthermore, there is a pair of binary 1's in this column. Thus, a binary 1 carry is generated into the third column. At this point in the process you have:

Carry: 1
Value #1: 1011
Value #2: + 1110
Sum: 01

Next, go to the third column. Here, you find two binary 1's consisting of the binary 1 carry generated by the second column, and a binary 1 from Value #2. Since two is even, the sum of the third column is 0. Also, there is a pair of binary 1's in this column. Consequently, a binary 1 carry is generated into the fourth column. At this point you have:

Carry: 11
Value #1: 1011
Value #2: + 1110
Sum: 001

Going to the fourth column, you find three 1's, including the carry. Since three is odd, the sum of this column is 1. Also, a carry must be generated into a fifth column, since there is one pair of 1's in the fourth column. Now you have:

Carry: 111
Value #1: 1011
Value #2: + 1110
Sum: 1001

Notice that the addition of column four has generated a **last carry** into an imaginary fifth column. This column must also be added. Since there is only one binary 1 in this column (the carry) the sum is 1. The final tabulation is:

Carry:	111
Value #1:	1011
Value #2:	+ 1110
Sum:	11001

Thus, 1011 + 1110 = 11001. To check this result, perform the addition in decimal. The binary value 1011 is equivalent to the decimal value 11, and the binary value 1110 is equivalent to the decimal value 14. Now, $11_{10} + 14_{10} = 25_{10}$. Since $25_{10} = 11001_2$, the above addition is correct.

Example A-9

Perform the following additions:

a. 1001 + 1110
b. 1010 + 0110 + 1111
c. 0101 + 1100 + 1011 + 1111

Applying the above binary addition steps you get:

a.

Carry:	1
Value #1:	1001
Value #2:	+ 1110
Sum:	10111

b.

Carry:	111
Value #1:	1010
Value #2:	0110
Value #3:	+ 1111
Sum:	11111

c.

Carry:	11 11111
Value #1:	0101
Value #2:	1100
Value #3:	1011
Value #4:	+ 1111
Sum:	101011

Notice that columns three and four in this problem both generate two carry's into the next column, since each of these columns contains **two pairs** of 1's. The two carry's are both placed in the next respective column, one above the other as shown.

Of course, more and larger binary values could be added, but the basic idea is the same. Now that you know how to add in binary, let's see how binary addition is used to subtract binary numbers.

Binary Subtraction

Subtraction is just a special case of addition. From high school algebra you know that to subtract, you simply take the negative of the value to be subtracted and add. For instance to subtract 3 from 5 you take the negative of 3, which is −3, then add it to 5 to get a result of 2. In symbols,

$$5 - 3 = 5 + (-3) = 2$$

This is exactly the way that digital computer circuits subtract binary numbers. You already know how to add two binary values. So, in order to subtract, all you need to learn is how to take the negative of a binary number.

The negative of a binary number is its **two's complement**. To find the two's complement of a binary value you must add 1 to its **one's complement**. The one's complement of a binary value is obtained by **inverting** all the bits in the value. For example, the one's complement of 0101 is 1010. In other words, the one's complement of a binary number is obtained by changing all the 1's to 0's and all the 0's to 1's.

Now back to two's complement. As I stated, the two's complement of a binary value is found by adding 1 to its one's complement. Therefore, the two's complement of 0101 is:

$$\begin{aligned}\text{Two's Complement} &= (\text{One's Complement} + 1)\\ &= \text{One's Complement of } 0101 + 1\\ &= 1010 + 1\\ &= 1011\end{aligned}$$

By definition, the negative of any binary number is its two's complement. So, from the example, the negative of 0101 is 1011. In other words, since $0101_2 = 5_{10}$, then $1011_2 = -5_{10}$ in two's complement.

One final point, zero is always represented in two's complement as a positive value using multiple 0's. For example, the 8-bit two's complement representation for zero is 0000 0000.

Example A-10

Express each of the following decimal numbers as a 4-bit two's complement binary value:

a. -1_{10}
b. -2_{10}
c. -7_{10}

The negative of a binary value is its two's complement. The two's complement of a binary value is found by adding 1 to its one's complement. The first step is to convert the decimal value to a positive binary value. Then determine its one's complement.

a. $+1_{10} = 0001_2$

One's Complement of 0001 = 1110
Two's Complement of 0001 = 1110 + 1
= 1111

Thus, $-1_{10} = 1111_2$.

b. $+4_{10} = 0010_2$

One's Complement of 0010 = 1101
Two's Complement of 0010 = 1101 + 1
= 1110

Thus, $-4_{10} = 1110_2$.

c. $+7_{10} = 0111_2$

One's Complement of 0111 = 1000
Two's Complement of 0111 = 1000 + 1
= 1001

Thus, $-7_{10} = 1001_2$.

To subtract, you simply add the two's complement of the binary value to be subtracted. For instance, suppose that you must subtract 3 from 5 using 4-bit two's complement code. The 4-bit equivalents of 3 and 5 are 0011 and 0101, respectively. Since 0011 is to be subtracted from 0101, you must add the two's complement of 0011 to 0101. The two's complement of 0011 is 1 + 1100, or 1101. Therefore,

$$5 - 3 = 5 + (-3) = 0101 + 1101$$

or,

$$\begin{array}{rcr} & & 1101 \quad \text{(Carry's)} \\ 5 & = & 0101 \\ \underline{+\ (-3)} & = & \underline{+\quad 1101} \\ 2 & = & \underset{\uparrow}{\underline{1}}0010 \end{array}$$

Disregard Last Carry

Notice that the addition results in a carry into the last column. When subtracting, any **last carry** must **always** be disregarded to obtain the correct result.

Example A-11

Perform the following subtraction operations using 4-bit two's complement. Check your results by converting to decimal.

a. 7 − 4
b. 6 − 2
c. 4 − 7

To subtract in binary, you must add the two's complement of the value to be subtracted and disregard any last carry in the result. Using this idea produces the following solutions:

a. $7_{10} = 0111_2$
$4_{10} = 0100_2$

$$\begin{aligned} 7 - 4 &= 7 + (-4) \\ &= 0111 + (\text{Two's Complement of } 0100) \\ &= 0111 + (1011 + 1) \\ &= 0111 + 1100 \\ &= \underset{\uparrow}{1}0011 \end{aligned}$$

Disregard

$$\begin{aligned} &= 0011_2 \\ &= 3_{10} \quad \text{(Check)} \end{aligned}$$

b. $6_{10} = 0110_2$
$2_{10} = 0010_2$

$6 - 2 = 6 + (-2)$
$= 0110 + (\text{Two's Complement of } 0010)$
$= 0110 + (1101 + 1)$
$= 0110 + 1110$
$= \underset{\llcorner\text{Disregard}}{1}0100$
$= 0100_2$
$= 4_{10}$ (Check)

c. $4_{10} = 0100_2$
$7_{10} = 0111_2$

$4 - 7 = 0100 + (\text{Two's Complement of } 0111)$
$= 0100 + (1000 + 1)$
$= 0100 + 1001$
$= 1101_2$

When the preceding two's complement binary addition is performed, there is no last carry generated into the last column. Thus, the final result is 1101. However, is this correct? The decimal result must be -3. So, the question is: Does $-3 = 1101$? Since $+3 = 0011$, $-3 = 1100 + 1 = 1101$ which checks with the result. Remember that a negative binary number is always expressed in its two's complement form.

SELF–TEST REVIEW

1. Decimal, binary, and hex are all ______________ number systems.

2. Convert the following binary numbers to decimal:

    A. 1100 __________________
    B. 10 0110 __________________
    C. 1011 0111 __________________
    D. 10 0101 1111 __________________

3. Convert the following decimal numbers to binary:

    A. 13 __________________
    B. 39 __________________
    C. 184 __________________
    D. 608 __________________

4. A light emitting diode, or LED, is an electronic display device that is used to display binary data. A binary 1 is indicated when it is on, or illuminated, and a binary 0 is indicate when it is off, or not illuminated. Suppose a digital circuit uses eight LEDs as shown in Figure A-2 to display its output data. Assume that a dark circle means a given LED is on, while a white circle means that it is off.

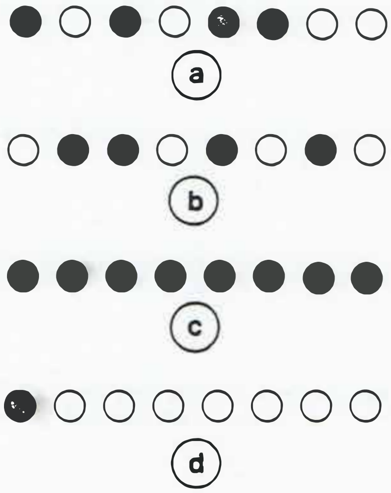


Figure A-2
LED display for Question 4.

A. What binary value is being displayed by each output?

a. ____________
b. ____________
c. ____________
d. ____________

B. What is the decimal equivalent of each display?

a. ____________
b. ____________
c. ____________
d. ____________

C. What is the hexadecimal equivalent of each display?

a. ____________
b. ____________
c. ____________
d. ____________

5. How many bits, bytes, and nibbles are in the following numbers?

A. $0110\ 1101_2$ ____________, ____________, ____________
B. $1011\ 1000\ 0000\ 1111_2$ ____________, ____________, ____________
C. $2FC_{16}$ ____________, ____________, ____________

6. Suppose an 8-bit LED output is used to display a binary count from 0000 0000 to 1111 1111.

A. What is the maximum count value, in decimal, that can be displayed?
____

B. How many unique count displays are there? ____________

7. A personal computer has a 128K byte user memory.

A. How many bytes of user memory does it have? ____________
B. How many bits of user memory does it have? ____________

8. If a digital system has 32,768 bits of memory, how many K-bytes of memory does it have? ____________

9. Convert the following numbers to binary:

A. 72_{10} ____________
B. 72_{16} ____________
C. 135_{10} ____________
D. $4CF5_{16}$ ____________

10. A digital counting circuit has a hexadecimal output display that counts from 00_{16} to FF_{16}. Suppose that the display is presently indicating a count of 2A. What are the next six successive display outputs? ____________, ____________, ____________, ____________, ____________, and ____________.

11. Convert each of the following to hexadecimal:

A. $1110\ 1101_2$ ____________
B. $2{,}048_{10}$ ____________
C. $41{,}687_{10}$ ____________
D. $1011\ 0000\ 1111\ 0010_2$ ____________

12. A personal computer has 64K bytes of memory. Each byte of memory is assigned a hexadecimal address so that its contents can be accessed. Suppose the addresses begin at 0000_{16} and run up consecutively for each of the 64K bytes of memory.

A. What is the last hexadecimal memory address? ____________
B. What is the decimal equivalent of the last memory address? ____________
C. How many unique addresses, in decimal, are there? ____________

13. Perform the following binary additions (assume that all values are positive):

A. 0111 + 1011 = ____________
B. 0110 0011 + 1001 0110 = ____________
C. 0101 + 1110 + 1111 = ____________

14. Determine the one's complement of the following binary values:

A. 0000 ______________
B. 0101 ______________
C. 0011 1100 ______________

15. Determine the two's complement of the values in Question #14.

A. ______________
B. ______________
C. ______________

16. Express each of the following decimal numbers as a 4-bit binary value in two's complement:

A. −1 ______________
B. −2 ______________
C. −6 ______________

17. Perform the following subtraction operations using 4-bit two's complement. Check your results by converting to decimal.

A. 7 − 1 = ______________$_2$ and ______________$_{10}$
B. 3 − 2 = ______________$_2$ and ______________$_{10}$
C. 5 − 6 = ______________$_2$ and ______________$_{10}$

ANSWERS

1. weighted

2A. 12

2B. 38

2C. 183

2D. 607

3A. 1101

3B. 10 0111

3C. 1011 1000

3D. 10 0110 0000

4A.
- a. 1010 1100
- b. 0110 1010
- c. 1111 1111
- d. 1000 0000

4B.
- a. 172
- b. 106
- c. 255
- d. 128

4C.
- a. AC
- b. 6A
- c. FF
- d. 80

5A. 8 bits, 1 byte, 2 nibbles

5B. 16 bits, 2 bytes, 4 nibbles

5C. 12 bits, 1-1/2 bytes, 3 nibbles

6A. 255

6B. 256, including 0

7A. 128 × 1024 or 131,072

7B. 131,072 × 8 or 1,048,576

8. 32,768/8/1024 = 4K

9A. 0100 1000

9B. 0111 0010

9C. 1000 0111

9D. 0100 1100 1111 0101

Answers (continued)

10. 2B, 2C, 2D, 2E, 2F, and 30

11A. ED

11B. 800

11C. A2D7

11D. B0F2

12A. FFFF

12B. 65,535

12C. 65,536, including 0

13A. 10010

13B. 1111 1001

13C. 10 0010

14A. 1111

14B. 1010

14C. 1100 0011

15A. 0000

15B. 1011

15C. 1100 0100

16A. 1111

16B. 1110

16C. 1010

17A. $7 - 1 = 7 + (-1)$
$= 0111 + 1111$
$= \underset{\uparrow\text{—Disregard}}{1}0110$
$= 0110_2$
$= 6_{10}$ (check)

17B. $3 - 2 = 3 + (-2)$
$= 0011 + 1110$
$= \underset{\uparrow\text{—Disregard}}{1}0001$
$= 0001_2$
$= 1_{10}$ (check)

17C. $5 - 6 = 5 + (-6)$
$= 0101 + 1010$
$= 1111_2$
$= -1_{10}$ (check)

APPENDIX B

BASIC ELECTRONIC CONCEPTS

CONTENTS

Introduction . . . . Page B-3

DC Circuit Basics . . . . Page B-3

AC Circuit Basics . . . . Page B-11

Semiconductor Basics . . . . Page B-18

Self-Test Review . . . . Page B-23

Answers . . . . Page B-25

INTRODUCTION

This Appendix has been written to provide you with the electronic circuit background that you will need to complete this course. These electronic principles will be discussed in only their most basic and functional terms. If you already have a background in electronic circuit theory, you might want to just "skim" over this material and go directly to Unit 1. However, if you are unfamiliar with electronic circuits, you should study this material and work the related problems at the end of the Appendix. For an in-depth study of electronic circuit theory, I recommend the other electronic circuit design courses in this Engineering Design Series.

DC CIRCUIT BASICS

The abbreviation "DC" stands for **direct current**. By direct current, I mean a constant current that does not change with time. Of course, your probably wondering: What is current? For the purpose of this course, I will define current as the **movement of electrons** within a conductor.

A conductor is any material that will readily allow current to pass through it. Most metals such as copper, aluminum, and silver are good conductors. On the other hand, materials such as rubber, plastic, glass, and dry wood are not good conductors. Therefore, these materials are called insulators.

Current

So, current has been defined as the movement of electrons through a conductor. The next logical question is: How do the electrons move, and what makes them move? Electrons move within a conductor by moving from one conductor atom to the next. They are forced to move by placing a **difference in potential** across the conductor as shown in Figure B-1. Here, the potential difference is supplied by a DC potential source connected to the conductor as shown. One side of the source supplies a constant positive (+) potential to one end of the conductor, and the other side of the source supplies a constant negative (−) potential to the opposite end of the conductor.

The negative potential repels electrons at one end of the conductor, since like charges repel. The positive potential attracts electrons at the other end of the conductor, since unlike charges attract. The result of this potential difference across the conductor is electron movement through the conductor, from the negative side to the positive side of the the DC source. As long as the circuit remains connected, or closed, this electron movement continues. The electron movement ceases when the circuit is broken, or opened, at some point.

The electron flow within the circuit in Figure B-1 is the circuit current and is labeled I. Current is measured in **amperes**, abbreviated **amps**, or **A**. Simple digital circuits do not require more than one amp to operate. Thus, you will see terms such as **milliamps, microamps,** and **nanoamps** associated with digital circuits. Notice that the term "amp" is prefixed with milli, micro, or nano. Table B-1 lists the meaning of these prefixes. For example, a milliamp, or mA, is one-thousandth of an amp, or 1 milliamp = 0.001 amp. Also, using scientific notation, 1 milliamp = 1 × 10^{-3} amp.

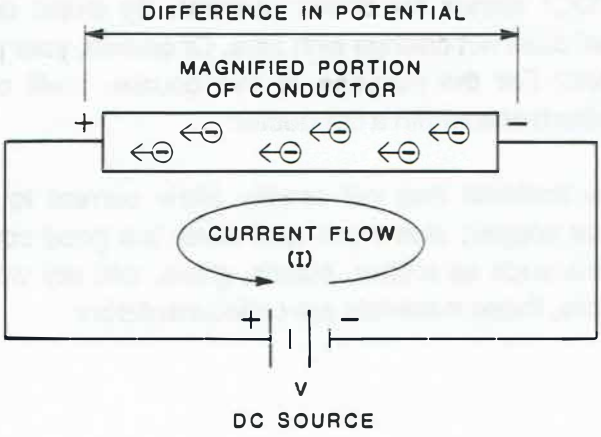


Figure B-1

Current flow (I) within a conductor is created by a difference in potential (V) across the conductor.

Example B-1

How many amps do each of the following quantities represent?

a. 3 mA
b. 5 μA
c. 10 nA

Using Table B-1, you can interpret these quantities as follows:

A. 3 mA = 3 milliamps = 3 × 10^{-3} amps = 0.003 amps.
B. 5 μA = 5 microamps = 5 × 10^{-6} amps = 0.000005 amps.
C. 10 nA = 10 nanoamps = 10 × 10^{-9} amps = 0.00000001 amps.

Table B-1: Common Prefixes Used With Electronic Terms

Prefix	Abbreviation	Decimal Equivalent	Scientific Notation
milli	m	0.001	1×10^{-3}
micro	μ	0.000001	1×10^{-6}
nano	n	0.000000001	1×10^{-9}
kilo	k	1000	1×10^{3}
mega	M	1000000	1×10^{6}
giga	G	1000000000	1×10^{9}

Voltage

The potential difference supplied by the DC source in Figure B-1 is the source **voltage** and is labeled **V**. Thus, a voltage is defined as a **difference in potential** within an electrical circuit. In other words, a voltage is seen any place where a difference in potential occurs within the circuit. Voltage is measured in **volts**, abbreviated **V**. Most digital circuits operate on voltages anywhere between 0 V and 25 V. By the way, a 0 V potential is called a **ground potential**.

Like current, voltages less than 1 V are abbreviated using the prefixes in Table B-1. For instance, 5 millivolts, or 5 mV, is equivalent to 0.005 V or 5×10^{-3} V.

Resistance

The simple circuit in Figure B-1 can be drawn as shown in Figure B-2. Here, the conductor in Figure B-1 has been replaced with a resistor symbol. A **resistor** is a physical device that offers resistance to the flow of current.

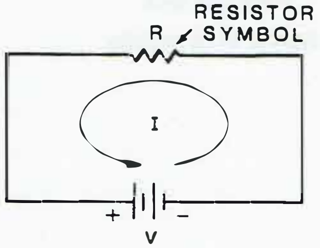


Figure B-2

A simple DC circuit consists of a resistor (R), current (I), and voltage (V).

The resistance of a resistor is measured in **Ohms**, abbreviated with the greek symbol Ω (omega). Most resistances used in digital circuits range from 100 Ω to 1,000,000 Ω. As a result, you will commonly see terms such as kilohms (kΩ) and megaohms (MΩ) associated with resistances. The prefixes kilo (k) and mega (M) have the meaning of 1,000 and 1,000,000, respectively (See Table B-1). Thus, a 10 kΩ resistor has a resistance of 10,000 ohms, or 10 × 10^3 ohms.

There are two types of resistors that you will run across in digital circuits: **fixed** resistors and **variable** resistors. A fixed resistor has a resistance value that remains fixed, or constant. These resistors have two leads and are normally made from carbon as shown in Figure B-3. There are color bands around the outside of the resistor as shown in Figure B-4 that indicate its fixed value. The associated table in Figure B-4 shows you how to read the color coding. Notice that resistors can have four or five color bands. If the resistor has four color bands, the first two bands indicate the first two numbers of resistor value. The third band indicates the number of zeros that must be added to the first two numbers. The last band indicates the amount of tolerance associated with the resistor. Most of the resistors you will encounter have only four color bands. If the resistor has five color bands, the first three bands represent the first three numbers of the value. Then the fourth and fifth bands indicate the number of zeros and tolerance, respectively.

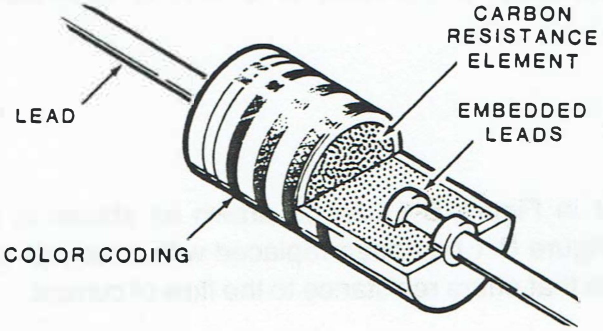
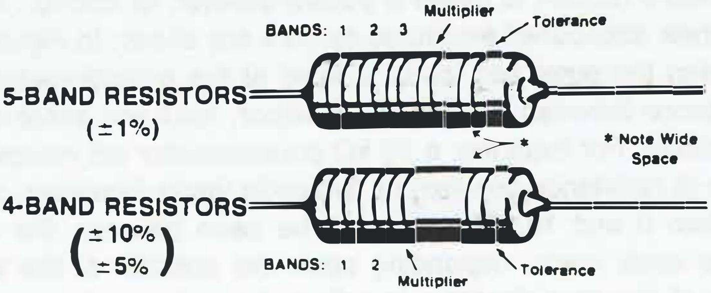


Figure B-3
A typical fixed resistor.



Band 1 1st Digit	
Color	Digit
Black	0
Brown	1
Red	2
Orange	3
Yellow	4
Green	5
Blue	6
Violet	7
Gray	8
White	9

Band 2 2nd Digit	
Color	Digit
Black	0
Brown	1
Red	2
Orange	3
Yellow	4
Green	5
Blue	6
Violet	7
Gray	8
White	9

Band 3 (if used) 3rd Digit	
Color	Digit
Black	0
Brown	1
Red	2
Orange	3
Yellow	4
Green	5
Blue	6
Violet	7
Gray	8
White	9

Multiplier	
Color	Multiplier
Black	1
Brown	10
Red	100
Orange	1.000
Yellow	10.000
Green	100.000
Blue	1.000.000
Silver	0.01
Gold	0.1

Resistance Tolerance	
Color	Tolerance
Silver	± 10%
Gold	± 5%
Red	± 2%
Brown	± 1%
Green	± .5%
Blue	± .25%
Violet	± .1%
Gray	± .05%

Figure B-4

The resistor color code.

Example B-2

Determine the value of a resistor that has color bands of brown, black, orange, silver.

The first two colors are brown and black. This means that the first two digits in the resistor value are 10 (See Figure B-4). Since the third color is orange, three 0's must be added to the first two digits. Thus, the resistor value is 10,000 ohms, or 10 kΩ. The fourth band is silver. This means that the resistor has a tolerance of ± 10%. As a result, the actual resistor value can range from 9,000 ohms to 11,000 ohms. This range is determined by taking 10% of 10,000 to get 1,000. Then 1,000 is subtracted from 10,000 to get 9,000 and added to 10,000 to get 11,000.

A variable resistor is called a **potentiometer**, or control. Typical potentiometers and their associated electronic symbol are shown in Figure B-5. The resistance between the outer two leads is fixed at the potentiometer value. However, the resistance between the middle, or **wiper**, lead and either one of the outer leads is variable. For instance, a 10 kΩ potentiometer will measure a constant 10,000 ohms of resistance between its two outer leads. However, a resistance anywhere between 0 and 10,000 ohms will be seen between the wiper and either one of the outer leads, depending upon the position of the wiper. The resistance value of the potentiometer is adjusted by turning the wiper shaft or disk. The value of a given potentiometer is usually stamped or printed somewhere on the device.

Both fixed and variable resistors are also rated by the amount of power they can dissipate. Typical power ratings for fixed resistors used in digital circuits are 1/4 watt, 1/2 watt, and 1 watt. Typical potentiometer power ratings range from 0.1 watt to 10 watts, depending on the physical size of the device. A **watt** of power is dissipated, in heat, when a 1 volt potential difference causes 1 amp of current to flow through the resistance. More about power later.

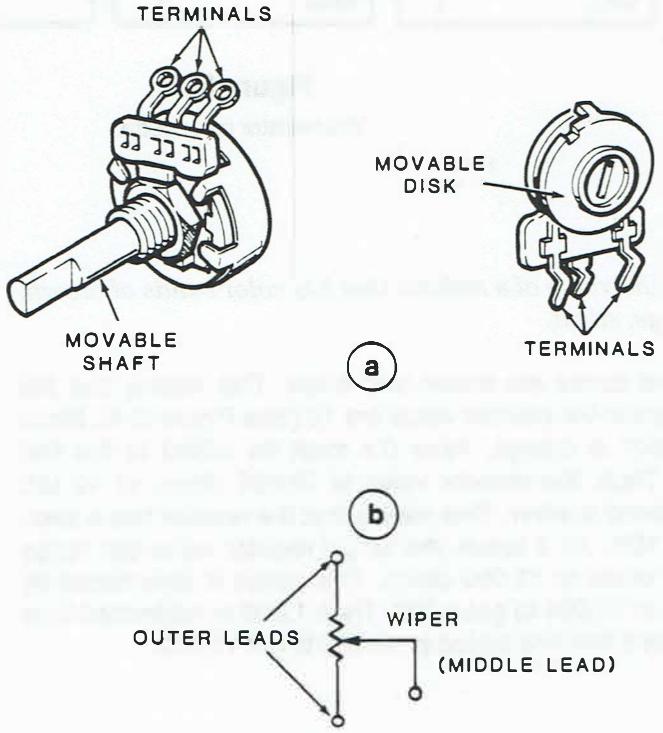


Figure B-5

Typical potentiometers (a) and electronic symbol (b).

Ohm's Law

At this point, you are familiar with the three basic quantities in an electrical circuit: current, voltage, and resistance. These three quantities are related to each other by an important relationship called **Ohm's Law**. Ohm's Law simply states that voltage equals current times resistance. In symbols,

$$V = I \times R, \text{ or } V = IR$$

Thus, by knowing two quantities and a little algebra, Ohm's Law allows you to calculate the third quantity.

Example B-3

Calculate the missing quantities in Figure B-6.

a. In Figure B-6a, the current and resistance are given, but the voltage is unknown. Applying Ohm's law directly, you get:

$$\begin{aligned} V &= IR \\ &= 1\ \text{mA} \times 10\ \text{k}\Omega \\ &= (1 \times 10^{-3}\ \text{A}) \times (10 \times 10^{3}\Omega) \\ &= 10\ \text{V} \end{aligned}$$

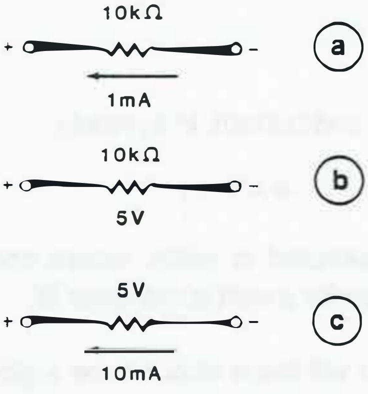


Figure B-6

Ohm's Law circuits for Example B-3.

b. The voltage and resistance are given in Figure B-6b, but the current is unknown. Using Ohm's Law and a little algebra gives you:

$$V = IR$$

Solving for I, you get:

$$\begin{aligned} I &= V/R \\ &= 5\ V/10\ k\Omega \\ &= 5\ V/(10 \times 10^{3}\Omega) \\ &= 0.5 \times 10^{-3}\ A \\ &= 0.5\ mA \end{aligned}$$

c. The current and voltage are known in Figure B-6c, but the resistance is unknown. Again with Ohm's law and a little algebra you get:

$$V = IR$$

Solving for R gives you:

$$\begin{aligned} R &= V/I \\ &= 5\ V/10\ mA \\ &= 5\ V/(10 \times 10^{-3}\ A) \\ &= 0.5 \times 10^{3}\Omega \\ &= 0.5\ k\Omega \end{aligned}$$

Power

Power is the product of voltage and current. In symbols,

$$P = V \times I$$

As stated earlier, power is measured in watts, where one watt equals one volt times one amp. The abbreviation for a watt is the letter **W**.

Most of the digital ICs that you will learn about have a **power dissipation** value associated with them. The power dissipation of an IC is the maximum amount of power that the IC is expected to consume under normal operating conditions. This value is important when designing digital circuits, since the power is dissipated directly in heat. As a result, special precautions must be taken to eliminate the dissipated heat so that the circuit does not become overheated.

You will commonly see terms like milliwatts and microwatts associated with digital devices. The prefixes milli and micro take on the same meaning as before (See Table B-1).

Example B-4

A digital IC data sheet specifies a maximum power dissipation of 10 mW and a supply voltage of 5 V. Calculate the amount of current that the IC will conduct.

Here, you are given power and voltage and must find the current. Solving the above power equation for current gives you:

$$
\begin{aligned}
I &= P/V \\
&= 10\text{ mW}/5\text{ V} \\
&= 10 \times 10^{-3}\text{ W}/5\text{ V} \\
&= 2 \times 10^{-3}\text{ A} \\
&= 2\text{ mA}
\end{aligned}
$$

AC CIRCUIT BASICS

The term AC stands for **alternating current**. An alternating current is one that changes with time as shown in Figure B-7. As you can see, there are various shapes of alternating current. The sine wave current shown in Figure 6-7a is the type that is available from your household receptacles. The square wave current shown in Figure 6-7d is the type most commonly used within digital circuits. Although digital circuits are primarily DC circuits, alternating square wave pulses like the one in Figure B-7d are used within the circuit to activate some of the digital devices.

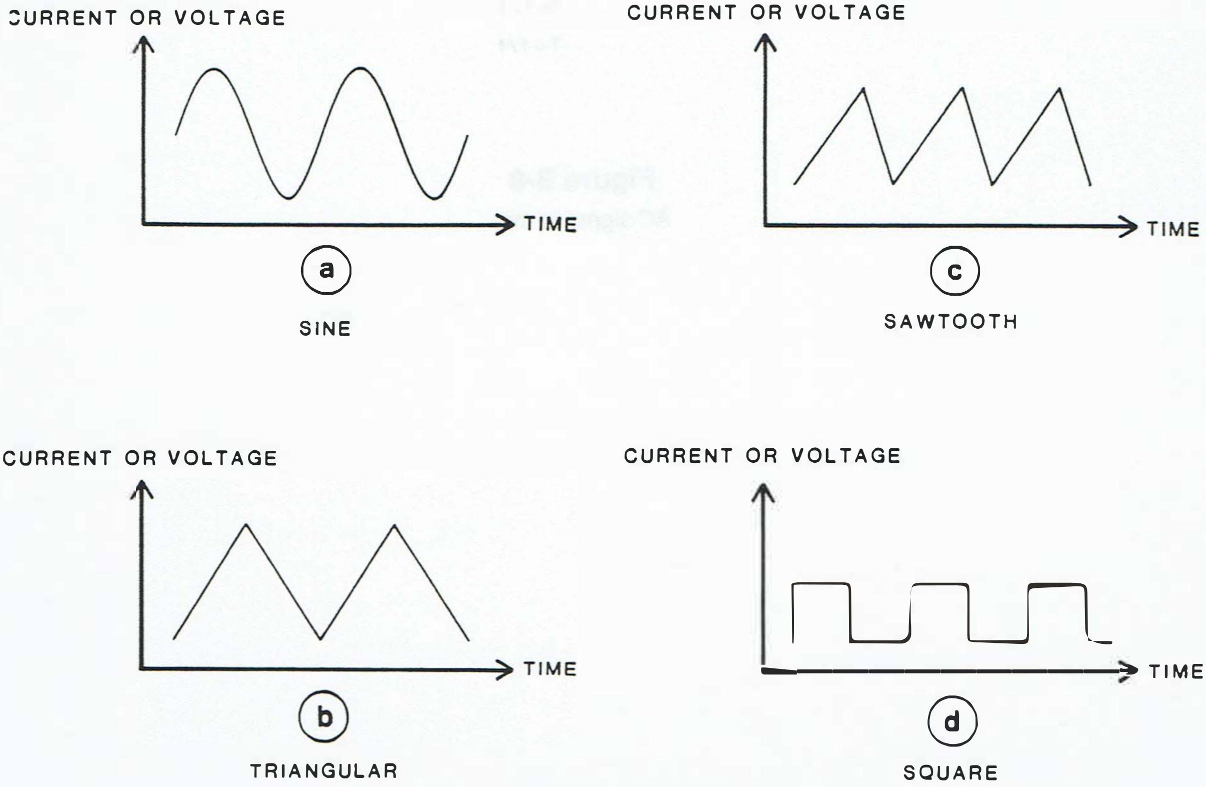
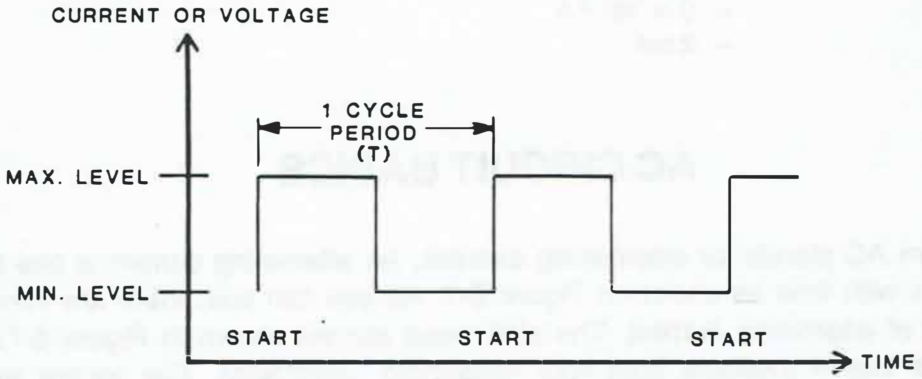


Figure B-7

Various types of AC waveforms.

Period and Frequency

The alternating signals that you will be using in digital design are **periodic** signals. This means that the signal alternates, or changes, at a fixed rate. Look at the alternating square wave signal in Figure B-8. Notice that the signal increases to a maximum level, then reverses itself and decreases to a minimum level. Then, it reverses itself and begins increasing again. One **cycle** is the interval between when the signal starts at some level, goes through its minimum and maximum levels and returns to the same starting level again.



T=PERIOD=LENGTH OF ONE CYCLE IN SECONDS

f=FREQUENCY=NUMBER OF CYCLES PER SECOND IN HERTZ (Hz)

$f = 1/T$

$T = 1/f$

Figure B-8

AC signal terms.

The length, or amount of time, of one cycle is called its **period** and is measured in **seconds**, or **S**. The **frequency** of the signal is the number of cycles that occur in one second and is measured in **Hertz**, or **Hz**. The symbol used to denote period is the upper case letter **T**. The symbol used to denote frequency is the lower case letter **f**.

Frequency and period are inversely proportional. In other words, as one goes up the other goes down. In symbols,

$$f = 1/T \text{ or } T = 1/f$$

Example B-5

A digital circuit uses a square wave signal that alternates at a rate of one million cycles per second.

a. What is the frequency of the signal?
b. What is the period of the signal?

a. Since frequency is the number of cycles in one second. The frequency must be 1 million Hertz, or 1 megahertz. In symbols,

$$f = 1 \text{ MHz}$$

Notice the use of the prefix mega, or M, which means one million (See Table B-1).

b. Period is inversely proportional to frequency, or:

$$\begin{aligned} T &= 1/f \\ &= 1/1 \text{ MHz} \\ &= 1/(1 \times 10^6) \text{ Hz} \\ &= 1 \times 10^{-6} \text{ second} \\ &= 1 \text{ microsecond} \\ &= 1\ \mu S \end{aligned}$$

Duty Cycle

You must be familiar with one final term that is associated with square wave signals. This term is **duty cycle**. The duty cycle of a square wave signal is the amount of time that the signal is maximum, or **high**, divided by the signal period, times 100%, or:

Duty Cycle = [(High Time)/(Period)] × 100%

In other words, the duty cycle of a signal is the percentage of time during the cycle that the signal is at its maximum, or high, level. Several different duty cycles are illustrated in Figure B-9. In Figure B-9a, the duty cycle is 50%, since the signal is high for one-half of the cycle period. The duty cycle is 25% in Figure B-9b, since the signal is high for one-quarter of the cycle period. Finally, the signal in Figure B-9c has a 75% duty cycle, since the signal is high 75% of the cycle period.

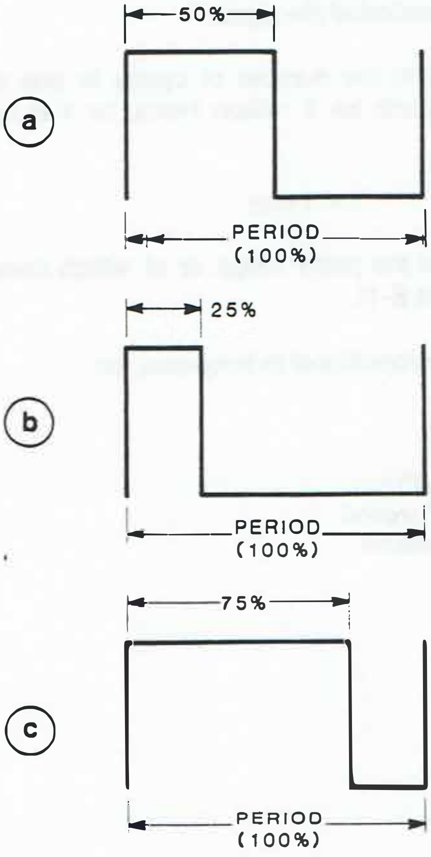


Figure B-9

Three different duty cycles: 50% (a), 25% (b), and 75% (c).

Capacitors

A capacitor is a physical device that is generally found in AC circuits. However, capacitors are also common in digital circuits. A capacitor consists of two conductive metal plates separated by an insulating material as shown in Figure B-10. Two leads are attached to the two metal plates as shown.

Capacitors are rated in **Farads**. The abbreviation for the Farad is the upper-case letter F. Most of the capacitors that you will be using in a digital circuit are in the microfarad (μF) and picofarad (pF) range.

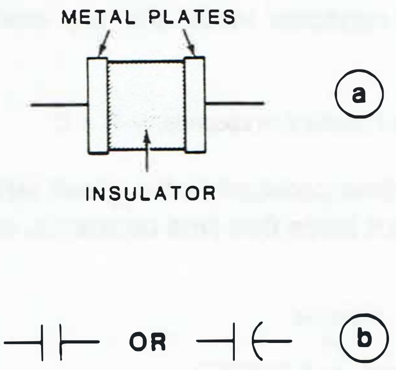


Figure B-10
A typical capacitor (a) and its electronic symbol (b).

Example B-6

A certain capacitor is valued at 0.000000000220 Farads.

- *a. What is the capacitor rating in microfarads?*
- *b. What is the capacitor rating in picofarads?*

a. One Farad equals one million, or 10^6, microfarads. Therefore,

$$\begin{aligned} 0.000000000220 \text{ Farads} &= 0.000000000220 \times 10^6 \text{ microfarads} \\ &= 0.000220 \text{ microfarads} \\ &= 0.000220\ \mu\text{F} \end{aligned}$$

b. One Farad equals one trillion, or 10^{12}, picofarads. Thus,

$$\begin{aligned} 0.000000000220 \text{ Farads} &= 0.000000000220 \times 10^{12} \text{ picofarads} \\ &= 220 \text{ picofarads} \\ &= 220 \text{ pF} \end{aligned}$$

The action of a capacitor in a DC circuit is illustrated by Figure B-11. Here, the capacitor has been placed in **series** with a DC voltage source, a switch, and a resistor. When the switch closes the capacitor "charges-up" to the DC source potential. In other words, after the switch is closed, the voltage potential across the capacitor (V_C) is equal to the DC source voltage (V). Current flow in the circuit lasts as long as the capacitor is charging. However, once it is charged, the capacitor **blocks** any more current flow.

As you can see from the associated charge curve, the capacitor does not charge-up to the source voltage potential instantaneously. A certain amount of time is required for it to become fully charged. The amount of charge time is related to something called **time constant**. The time constant of the charging circuit is simply the product of the capacitor value and any resistance in series with the capacitor. Thus,

$$\text{Time Constant, in seconds,} = R \times C$$

The symbol used to denote time constant is the greek letter τ (tau). Now, from Figure B-11, you can see that it takes **five** time constants, or 5 τ, for the capacitor to become fully charged.

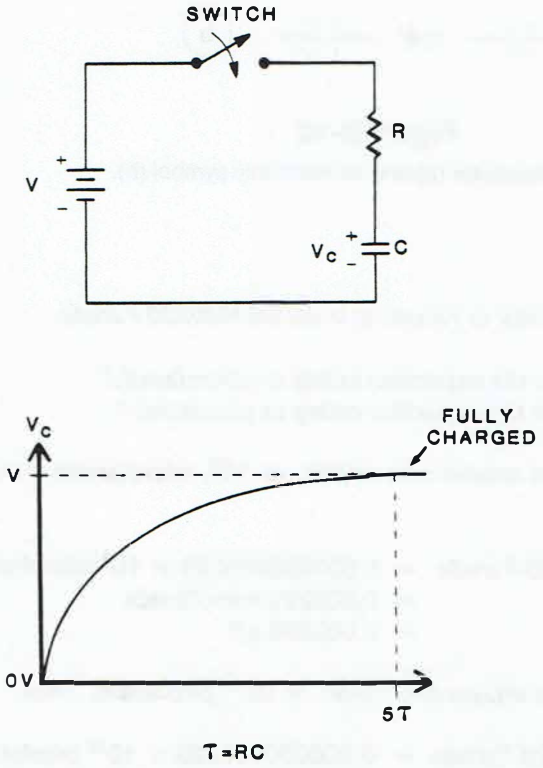


Figure B-11
Charging a capacitor.

Example B-7

A 0.01 μF capacitor is connected in series with a 10 kΩ resistance. How long will it take to completely charge the capacitor from a DC source.

The time constant of this charging circuit is:

$$\begin{aligned} \tau &= RC \\ &= 10\,k\Omega \times 0.01\,\mu F \\ &= (10 \times 10^{3})\,\Omega \times (.01 \times 10^{-6})F \\ &= 0.1 \times 10^{-3} \text{ seconds} \\ &= 0.1 \text{ mS} \end{aligned}$$

Now, it takes five time constants for the capacitor to become fully charged. Therefore,

$$\begin{aligned} \text{Charge time} &= 5\tau \\ &= 5 \times 0.1 \text{ mS} \\ &= 0.5 \text{ mS} \end{aligned}$$

Once a capacitor is charged, it can be discharged by placing it in a closed loop circuit like the one in Figure B-12. When the switch is closed, the voltage potential across the capacitor discharges to zero. How long does it take? You guessed it, five time constants. In other words, once the switch is closed, it take 5 × RC for the capacitor voltage (V_C) to discharge to zero. This can be seen by the associated discharge curve in Figure B-12.

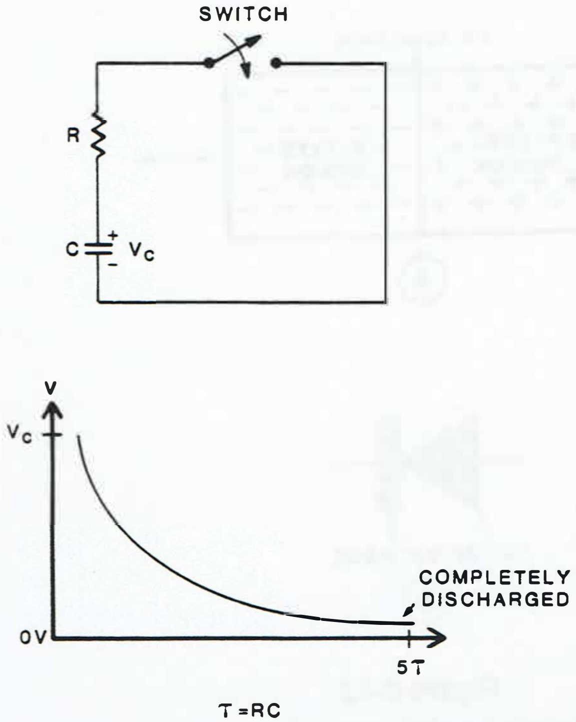


Figure B-12
Discharging a capacitor.

SEMICONDUCTOR BASICS

You need to be familiar with the fundamental operation of two semiconductor devices for simple digital designs: the **diode** and the **transistor**. It is only important that you understand the functional operation of these devices. For an in-depth study of diodes and transistors, consult the **Transistor Circuit Design** course in this Engineering Design Series.

The Diode

The diode is one of the simplest of semiconductor devices and its operation provides the basis for the operation of most all semiconductor devices. A diode is constructed by forming a **P-type** semiconductor region next to an **N-type** semiconductor region within a silicon or germanium chip (crystal wafer) as shown in Figure B-13a. The two semiconductor regions come together to form a **PN** junction as shown.

The N-type region has an excess number of electrons, while the P-type region has a deficiency of electrons. Leads are attached to both regions to allow for external connections to the device. The electronic symbol for a diode is shown in Figure B-13b. Observe that the P-type region is called the **anode** and the N-type region the **cathode** of the diode.

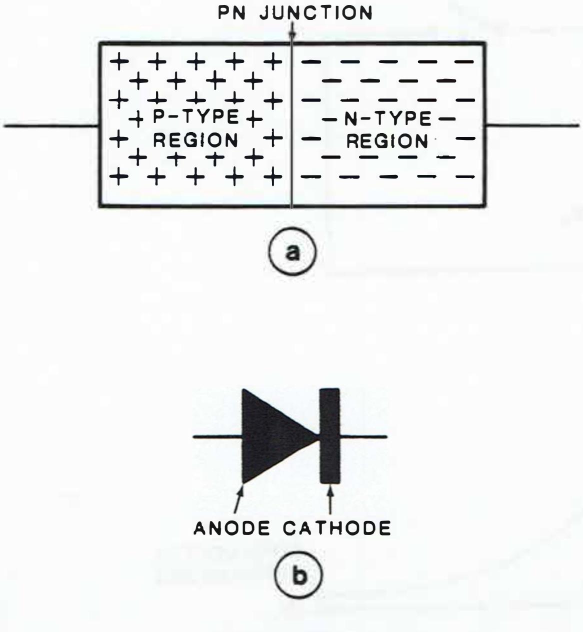


Figure B-13
The structure of a PN junction diode (a)
and its electronic symbol (b).

In its simplest form, a diode acts like a switch. When a positive potential is applied to the anode and a negative potential to the cathode, the diode conducts current. This is called **forward bias** and is illustrated by Figure B-14a. In this mode, the diode acts like a closed switch as shown. Now, look at Figure B-14b. Here, the bias potentials have been reversed, since a negative potential is applied to the anode and a positive potential to the cathode. For obvious reasons, this is called **reverse bias**. In this mode, the diode does not conduct current and, therefore, acts like an open switch. That's all there is to it!

I should mention that the biasing potentials do not have to be actual positive and negative potentials. With forward bias, the diode will conduct as long as the anode is about 1 V more positive, or higher, than the cathode. For instance, the diode will conduct when 5 V is applied to the anode and 0 V, or ground, is applied to the cathode. Likewise, to achieve reverse bias, the anode must be more negative, or lower, than the cathode. For example, the diode would [illegible] conduct if 0 V where applied to the anode and 5 V to the cathode.

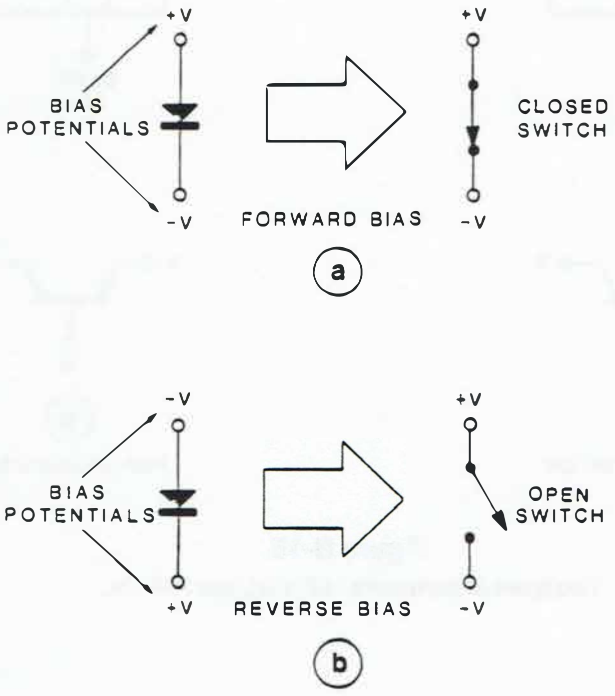


Figure B-14

The two operating modes of a diode:
forward bias (a) and reverse bias (b).

The Bipolar Transistor

Bipolar transistors are constructed by creating two PN junctions within a silicon or germanium chip as shown in Figure B-15. Notice that there are basically two ways to do this. Either a P-type region is sandwiched between two N-type regions, or an N-type region is sandwiched between two P-type regions. The first is called an **NPN transistor** and the second a **PNP transistor**.

Three leads are attached to each of the three regions as shown. The middle region is called the **base** and the two end regions are called the **emitter** and **collector** regions. Figure B-15 also shows the electronic symbols for both the NPN and PNP transistors. Notice that the emitter is designated with an arrow in the symbol. The arrow points to the base in the PNP symbol and points away from the base in the NPN symbol.

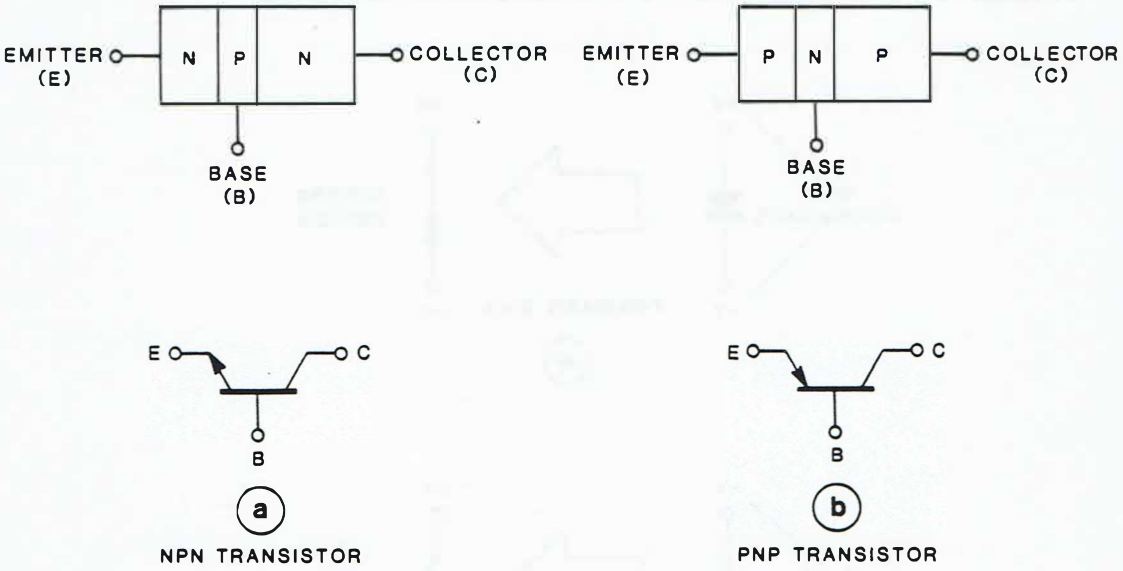


Figure B-15

Two types of transistors : NPN (a), and PNP (b).

In digital circuits, transistors are often used as **controlled switches** to turn something on or off. Here's how: An NPN transistor is shown in Figure B- 16 with its collector connected to a positive supply potential via a resistor and its emitter connected to ground. With no potential applied to the base, the transistor does not conduct and acts like an open switch. However, when a positive pulse is applied to the base as in Figure B-16b, the transistor conducts and acts like a closed switch for as long as the pulse is applied.

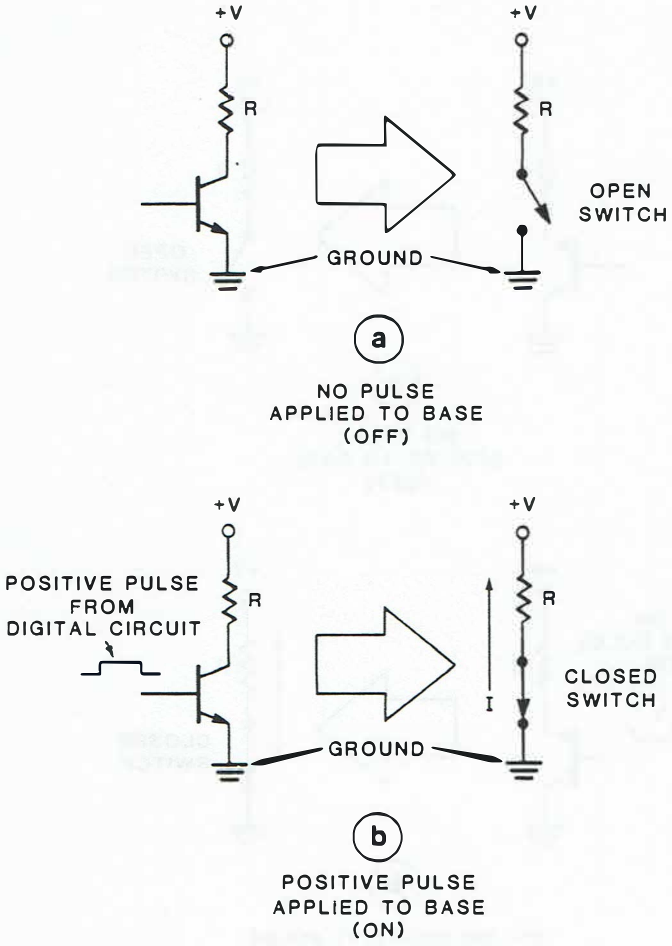


Figure B-16

A transistor is usually used as a controlled on/off switch in a digital circuit.

A PNP transistor works just the opposite of an NPN as shown in Figure B-17. Here, the collector is connected to ground and the emitter connected to the positive supply potential. Now the transistor switch closes when either a O V or negative pulse is applied to its base.

In digital designs, the digital circuit supplies the pulses to turn the transistor on and off. In this way, the digital circuit can **control** the operation of the transistor switch.

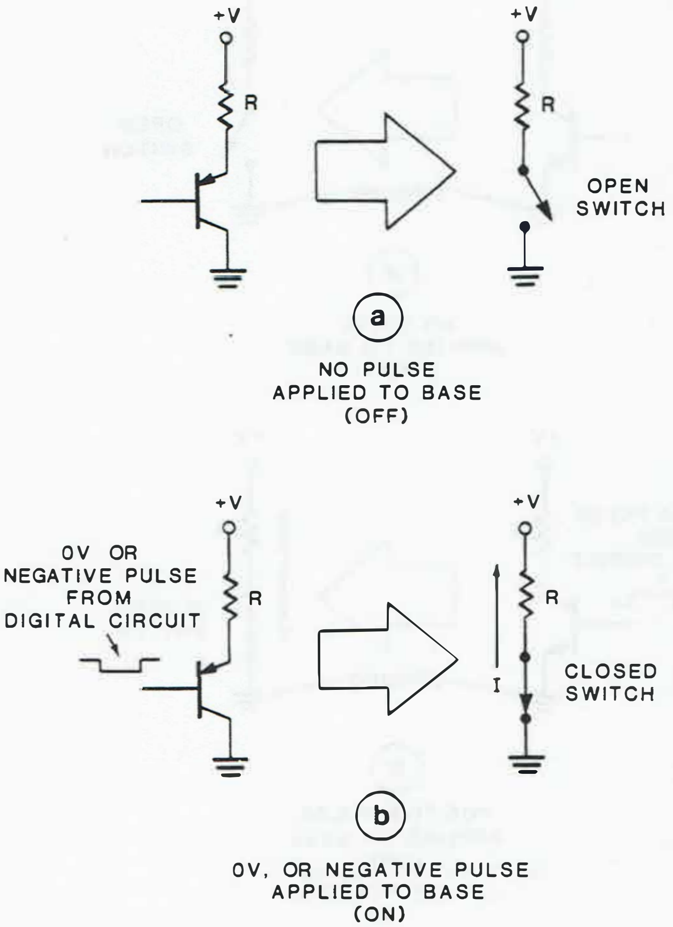

+V
R
+V
R
OPEN SWITCH
a
NO PULSE APPLIED TO BASE (OFF)
+V
0V OR NEGATIVE PULSE FROM DIGITAL CIRCUIT
R
+V
R
I
CLOSED SWITCH
b
0V, OR NEGATIVE PULSE APPLIED TO BASE (ON)

Figure B-17

A PNP transistor used as a digitally controlled switch.

SELF-TEST REVIEW

1. Define the following terms:

   A. Current ____________________

   B. Voltage ____________________

   C. Resistance ____________________

2. How many electrical units do the following quantities represent?

   A. 5 mA ____________ Amps

   B. 100 μV ____________ Volts

   C. 150 pF ____________ Farads

   D. 2.2 MΩ ____________ Ohms

   E. 100 kΩ ____________ Ohms

   F. 500 mW ____________ Watts

3. Determine the resistance and tolerance of the resistors with the following color coding:

   A. Brown, black, red, gold ____________

   B. Red, red, brown, silver ____________

   C. Orange, orange, brown, gold ____________

   D. Yellow, violet, red, brown, brown ____________

4. Use Ohm's Law to calculate the missing quantities below:

   A. 5 μA, 1 MΩ ____________

   B. 12 V, 10 kΩ ____________

   C. 20 mA, 5 V ____________

5. Use the power equation to calculate the missing quantities below:

   A. 20 mA, 5 V ____________

   B. 100 mW, 10 mA ____________

   C. 12 V, 1 W ____________

6. Define the following terms:

A. Period ______________________________
B. Frequency ______________________________
C. Duty Cycle ______________________________

7. The length of a square wave signal cycle is 0.0001 seconds. Find the following quantities:

A. The signal period equals ____________
B. The frequency of the signal equals ____________

8. What is the duty cycle of the signal in Question 7 if it is at its maximum, or high, level for 60 μs during each cycle? ____________

9. A certain capacitor is valued at 0.0000001 Farads. What is its rating in:

A. Microfarads ____________
B. Picofarads ____________

10. Suppose the capacitor in Question 9 is connected in series with a 10 MΩ resistance. How long will it take to completely charge the capacitor from a DC source? ____________

11. The two operating modes of a diode are ______________ bias and ______________ bias.

12. Explain how to make a diode conduct. ______________________
____
____

13. Explain how to make a diode act like an open switch. ____________
____
____

14. The two types of bipolar transistors are the ______________ and ______________.

15. Explain how to use an NPN bipolar transistor as a controlled on/off switch.
____
____
____
____

ANSWERS

1A. Current is the movement of electrons within a conductor.

1B. Voltage is a difference in potential within an electrical circuit.

1C. Resistance is the resistance to the flow of current offered by a resistor.

2A. 0.005 amps

2B. 0.0001 volts

2C. 0.00000000015 Farads

2D. 2,200,000 ohms

2E. 100,000 ohms

2F. 0.5 watts

3A. 1 kΩ ± 5%

3B. 220 Ω ± 10%

3C. 330 Ω ± 5%

3D. 4720 Ω ± 1%

4A. 5 V

4B. 1.2 mA

4C. 250 ohms

5A. 0.1 W

5B. 10 V

5C. 83.3 mA

6A. Period is the amount of time for one cycle.

6B. Frequency is the number of cycles in one second.

6C. Duty cycle is the amount of time that a square wave signal is high, divided by the signal period, times 100%.

7A. The signal period is 0.0001 sec., or 0.1 mS.

7B. The signal frequency is:

$$\begin{aligned} f &= 1/T \\ &= 1/0.1\ \text{mS} \\ &= 10\ \text{kHz} \end{aligned}$$

ANSWERS (continued)

8. 60%

9A. $0.0000001 \times 10^6 = 0.1\ \mu F$

9B. $0.0000001 \times 10^{12} = 100{,}000\ pF$

10. 5 sec

11. forward, reverse

12. A diode is made to conduct by applying a voltage potential to the anode that is about 1 V more positive than the cathode voltage potential.

13. You must reverse bias a diode to make it act like an open switch. This is done by applying a voltage potential to the anode that is more negative than the cathode voltage.

14. NPN, PNP

15. An NPN is made to act as a controlled on/off switch by connecting its collector to a positive supply potential via a resistor and its emitter to ground. With no potential applied to the base, the transistor does not conduct and acts like an open, or off, switch. When a positive pulse is applied to the base, the transistor conducts and acts like a closed, or on, switch for the duration of the pulse.

Solution to Questions 2, 4, 5, 8, and 10.

2A. 5 mA = 5 milliamps
= 5×10^{-3} amps
= 0.005 amps

2B. 100 μV = 100 microvolts
= 100×10^{-6} volts
= 0.0001 volts

2C. 150 pF = 150 picofarads
= 150×10^{-12} Farads
= 0.00000000015 Farads

2D. 2.2 MΩ = 2.2 megaohms
= 2.2×10^{6} ohms
= 2,200,000 ohms

2E. 100 kΩ = 100 kilohms
= 100×10^{3} ohms
= 100,000 ohms

2F. 500 mW = 500 milliwatts
= 500×10^{-3} watts
= 0.5 watts

4A. $V = IR$
= 5 μA × 1 MΩ
= $(5 \times 10^{-6}\,\text{A}) \times (1 \times 10^{6}\,\Omega)$
= 5 V

4B. $I = V/R$
= 12 V/10 kΩ
= $12\,\text{V}/(10 \times 10^{3}\,\Omega)$
= 1.2×10^{-3} A
= 1.2 mA

4C. $R = V/I$
= 5 V/20 mA
= $5\,\text{V}/(20 \times 10^{-3}\,\text{A})$
= $0.25 \times 10^{3}\,\Omega$
= 250 ohms

5A. $P = VI$
= 5 V × 20 mA
= $5\,\text{V} \times (20 \times 10^{-3}\,\text{A})$
= 100×10^{-3} watts
= 0.1 W

5B. $V = P/I$
= 100 mW/10 mA
= $(100 \times 10^{-3}\,\text{W})/(10 \times 10^{-3}\,\text{A})$
= 10×10^{0} V
= 10 V

5C. $I = P/V$

$= 1\ W/12\ V$

$= 0.0833\ A$

$= 83.3\ mA$

8. Duty Cycle $=$ [(High Time)/(Period)] $\times$ 100%

$= [60\ \mu S/0.1\ mS] \times 100\%$

$= [(60 \times 10^{-6})/(0.1 \times 10^{-3})] \times 100\%$

$= [600 \times 10^{-3}] \times 100\%$

$= 60\ \%$

10. $\tau = RC$

$= 10\ M\Omega \times 0.1\ \mu F$

$= (10 \times 10^{6}\Omega) \times (0.1 \times 10^{-6}\ F)$

$= 1 \times 10^{0}\ sec$

$= 1\ sec$

Charge time $= 5\tau$

$= 5 \times 1\ sec$

$= 5\ sec$

APPENDIX C

DATA SHEETS OF ICs USED IN EXPERIMENTS

CONTENTS

74LS00 Quad 2-Input NAND Gate ...... (Pin Assignment) .... Page C-4
(Electrical) .... Page C-10

74LS02 Quad 2-Input NOR Gate .... (Pin Assignment) .... Page C-4
(Electrical) .... Page C-12

74LS04 Hex Inverter .... (Pin Assignment) .... Page C-5
(Electrical) .... Page C-10

74LS10 Triple 3-Input NAND Gate ...... (Pin Assignment) .... Page C-6
(Electrical) .... Page C-10

74LS20 Dual 4-Input NAND Gate .... (Pin Assignment) .... Page C-7
(Electrical) .... Page C-10

74LS74 Dual D Flip-Flop .... (Pin Assignment) .... Page C-8
(Electrical) .... Page C-14

74LS76 Dual JK Flip-Flop .... (Pin Assignment) .... Page C-9
(Electrical) .... Page C-14

741 Operational Amplifier .... Page C-16

7447 7-Segment Decoder/Driver .... Page C-20

74LS86 Quad 2-Input XOR Gate .... Page C-25

74LS90 Decade Counter .... Page C-27

74LS138 Decoder/Demultiplexer .... Page C-33

74LS153 Multiplexer .... Page C-37

74LS193 4-Bit Up/Down Counter .... Page C-40

9402 Voltage-to-Frequency Converter .... Page C-47

ADC0804 Analog-to-Digital Converter .... Page C-55

INTRODUCTION

This appendix is a collection of data sheets for the integrated circuits used in the Experiments. They were furnished courtesy of **Texas Instruments, National Semiconductor,** and **Teledyne Semiconductor.** Each of the first seven ICs have two page references. The first is for the IC pin assignment, while the second is for the electrical characteristics. This is a system used by Texas Instruments to group the common elements of their simpler ICs.

54/74 FAMILIES OF COMPATIBLE TTL CIRCUITS

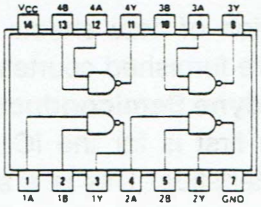
PIN ASSIGNMENTS (TOP VIEWS)

QUADRUPLE 2-INPUT POSITIVE-NAND GATES

00

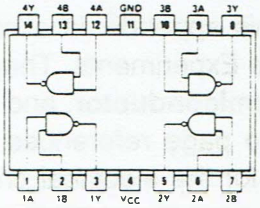
positive logic:
$Y = \overline{AB}$

See page 6-2



SN5400 (J)
SN54H00 (J)
SN54L00 (J)
SN54LS00 (J, W)
SN54S00 (J, W)

SN7400 (J, N)
SN74H00 (J, N)
SN74L00 (J, N)
SN74LS00 (J, N)
SN74S00 (J, N)

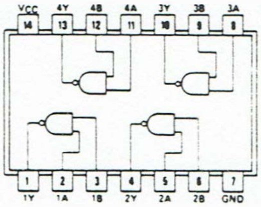


SN5400 (W)
SN54H00 (W)
SN54L00 (T)

QUADRUPLE 2-INPUT POSITIVE-NAND GATES WITH OPEN-COLLECTOR OUTPUTS

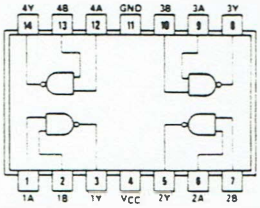
01

positive logic:
$Y = \overline{AB}$

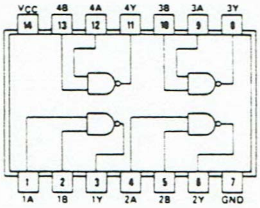


SN5401 (J)
SN54LS01 (J, W)

SN7401 (J, N)
SN74LS01 (J, N)



SN5401 (W)
SN54H01 (W)
SN54L01 (T)



SN54H01 (J) SN74H01 (J, N)

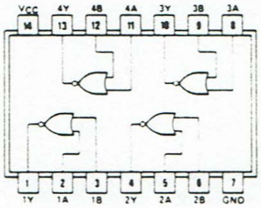
See page 6-4

QUADRUPLE 2-INPUT POSITIVE-NOR GATES

02

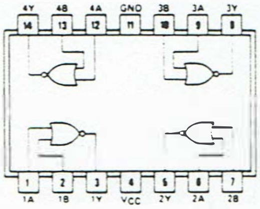
positive logic:
$Y = \overline{A+B}$

See page 6-8



SN5402 (J)
SN54L02 (J)
SN54LS02 (J, W)
SN54S02 (J, W)

SN7402 (J, N)
SN74L02 (J, N)
SN74LS02 (J, N)
SN74S02 (J, N)



SN5402 (W)
SN54L02 (T)

54/74 FAMILIES OF COMPATIBLE TTL CIRCUITS

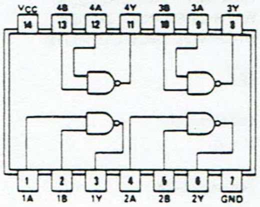
PIN ASSIGNMENTS (TOP VIEWS)

QUADRUPLE 2-INPUT
POSITIVE-NAND GATES
WITH OPEN-COLLECTOR OUTPUTS

03

positive logic:
$Y = \overline{AB}$

See page 6-4



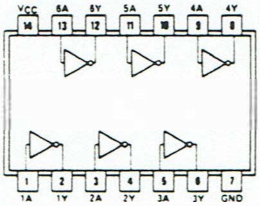
SN5403 (J)
SN54L03 (J)
SN54LS03 (J, W)
SN54S03 (J, W)
SN7403 (J, N)
SN74L03 (J, N)
SN74LS03 (J, N)
SN74S03 (J, N)

HEX INVERTERS

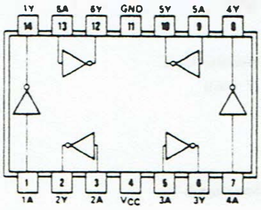
04

positive logic:
$Y = \overline{A}$

See page 6-2



SN5404 (J)
SN54H04 (J)
SN54L04 (J)
SN54LS04 (J, W)
SN54S04 (J, W)
SN7404 (J, N)
SN74H04 (J, N)
SN74L04 (J, N)
SN74LS04 (J, N)
SN74S04 (J, N)



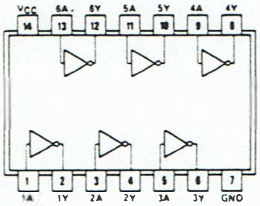
SN5404 (W)
SN54H04 (W)
SN54L04 (T)

HEX INVERTERS
WITH OPEN-COLLECTOR OUTPUTS

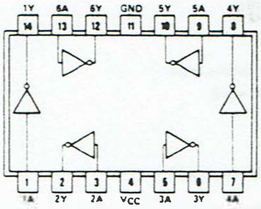
05

positive logic:
$Y = \overline{A}$

See page 6-4



SN5405 (J)
SN54H05 (J)
SN54LS05 (J, W)
SN54S05 (J, W)
SN7405 (J, N)
SN74H05 (J, N)
SN74LS05 (J, N)
SN74S05 (J, N)



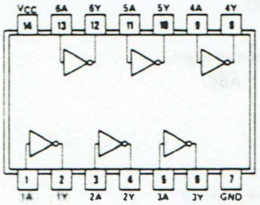
SN5405 (W)
SN54H05 (W)

HEX INVERTER BUFFERS/DRIVERS
WITH OPEN-COLLECTOR
HIGH-VOLTAGE OUTPUTS

06

positive logic:
$Y = \overline{A}$

See page 6-24



SN5406 (J, W)
SN7406 (J, N)

54/74 FAMILIES OF COMPATIBLE TTL CIRCUITS

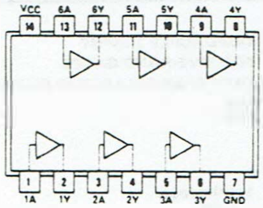
PIN ASSIGNMENTS (TOP VIEWS)

HEX BUFFERS/DRIVERS
WITH OPEN-COLLECTOR
HIGH-VOLTAGE OUTPUTS

07

positive logic:
Y = A

See page 6-24



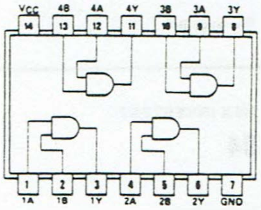
SN5407 (J, W) SN7407 (J, N)

QUADRUPLE 2-INPUT
POSITIVE-AND GATES

08

positive logic:
Y = AB

See page 6-10



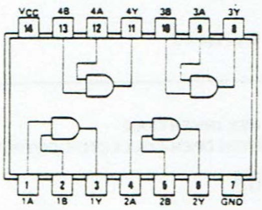
SN5408 (J, W) SN7408 (J, N)
SN54LS08 (J, W) SN74LS08 (J, N)
SN54S08 (J, W) SN74S08 (J, N)

QUADRUPLE 2-INPUT
POSITIVE-AND GATES
WITH OPEN-COLLECTOR OUTPUTS

09

positive logic:
Y = AB

See page 6-12



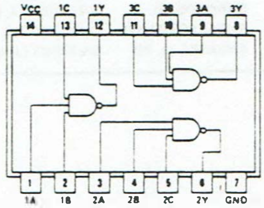
SN5409 (J, W) SN7409 (J, N)
SN54LS09 (J, W) SN74LS09 (J, N)
SN54S09 (J, W) SN74S09 (J, N)

TRIPLE 3-INPUT
POSITIVE-NAND GATES

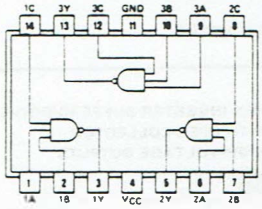
10

positive logic:
Y = $\overline{ABC}$

See page 6-2



SN5410 (J) SN7410 (J, N)
SN54H10 (J) SN74H10 (J, N)
SN54L10 (J) SN74L10 (J, N)
SN54LS10 (J, W) SN74LS10 (J, N)
SN54S10 (J, W) SN74S10 (J, N)



SN5410 (W)
SN54H10 (W)
SN54L10 (T)

54/74 FAMILIES OF COMPATIBLE TTL CIRCUITS

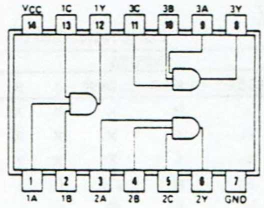
PIN ASSIGNMENTS (TOP VIEWS)

TRIPLE 3-INPUT
POSITIVE-AND GATES
WITH OPEN-COLLECTOR OUTPUTS

15

positive logic:
Y = ABC

See page 6-12



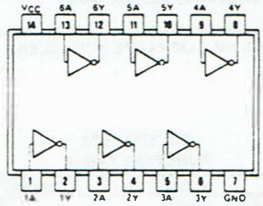
SN54H15 (J, W)	SN74H15 (J, N)
SN54LS15 (J, W)	SN74LS15 (J, N)
SN54S15 (J, W)	SN74S15 (J, N)

HEX INVERTER BUFFERS/DRIVERS
WITH OPEN-COLLECTOR
HIGH-VOLTAGE OUTPUTS

16

positive logic:
$Y = \bar{A}$

See page 6-24



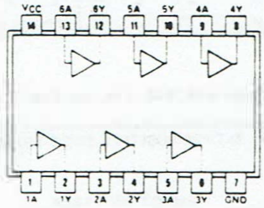
SN5416 (J, W) SN7416 (J, N)

HEX BUFFERS/DRIVERS
WITH OPEN-COLLECTOR
HIGH-VOLTAGE OUTPUTS

17

positive logic:
Y = A

See page 6-24



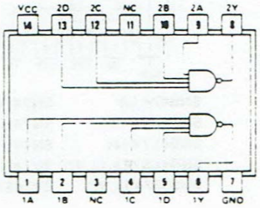
SN5417 (J, W) SN7417 (J, N)

DUAL 4-INPUT
POSITIVE-NAND GATES

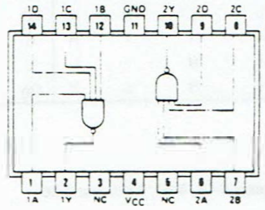
20

positive logic:
$Y = \overline{ABCD}$

See page 6-2



SN5420 (J)	SN7420 (J, N)
SN54H20 (J)	SN74H20 (J, N)
SN54L20 (J)	SN74L20 (J, N)
SN54LS20 (J, W)	SN74LS20 (J, N)
SN54S20 (J, W)	SN74S20 (J, N)



SN5420 (W)
SN54H20 (W)
SN54L20 (T)

NC – No internal connection

54/74 FAMILIES OF COMPATIBLE TTL CIRCUITS

PIN ASSIGNMENTS (TOP VIEWS)

AND-GATED J-K MASTER-SLAVE FLIP-FLOPS WITH PRESET AND CLEAR

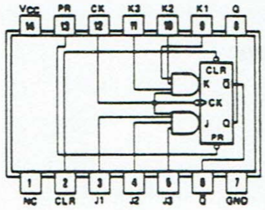
72

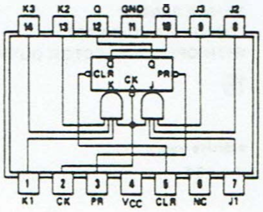
FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	⎍	L	L	Q_0	$\bar{Q}_0$
H	H	⎍	H	L	H	L
H	H	⎍	L	H	L	H
H	H	⎍	H	H	TOGGLE	

positive logic: J = J1·J2·J3; K1·K2·K3

See pages 6-46, 6-50, and 6-54





SN5472 (J)	SN7472 (J, N)	SN5472 (W)
SN54H72 (J)	SN74H72 (J, N)	SN54H72 (W)
SN54L72 (J)	SN74L72 (J, N)	SN54L72 (T)

NC—No internal connection

DUAL J-K FLIP-FLOPS WITH CLEAR

73

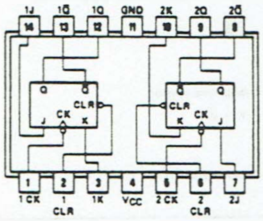
'73, 'H73, 'L73
FUNCTION TABLE

INPUTS				OUTPUTS	
CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	X	X	X	L	H
H	⎍	L	L	Q_0	$\bar{Q}_0$
H	⎍	H	L	H	L
H	⎍	L	H	L	H
H	⎍	H	H	TOGGLE	

'LS73A
FUNCTION TABLE

INPUTS				OUTPUTS	
CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	X	X	X	L	H
H	↓	L	L	Q_0	$\bar{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	TOGGLE	
H	H	X	X	Q_0	$\bar{Q}_0$

See pages 6-46, 6-50, 6-54, and 6-56



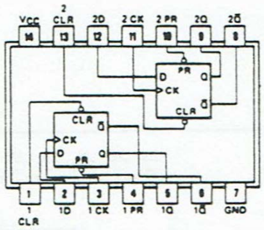
SN5473 (J, W)	SN7473 (J, N)
SN54H73 (J, W)	SN74H73 (J, N)
SN54L73 (J, T)	SN74L73 (J, N)
SN54LS73A (J, W)	SN74LS73A (J, N)

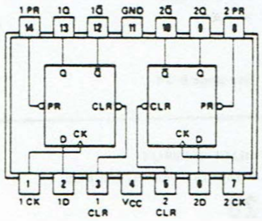
DUAL D-TYPE POSITIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

74

FUNCTION TABLE

INPUTS				OUTPUTS	
PRESET	CLEAR	CLOCK	D	Q	$\bar{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\bar{Q}_0$





SN5474 (J)	SN7474 (J, N)	SN5474 (W)
SN54H74 (J)	SN74H74 (J, N)	SN54H74 (W)
SN54L74 (J)	SN74L74 (J, N)	SN54L74 (T)
SN54LS74A (J, W)	SN74LS74A (J, N)	
SN54S74 (J, W)	SN74S74 (J, N)	

See pages 6-46, 6-50, 6-54, and 6-56

See explanation of function tables on page 3-8.

*This configuration is nonstable; that is, it will not persist when preset or clear inputs return to their inactive (high) level. Furthermore, the output levels of the 'LS74A in this configuration are not guaranteed to meet the minimum levels for V_{OH} if the lows at preset and clear are near V_{IL} maximum.

54/74 FAMILIES OF COMPATIBLE TTL CIRCUITS

PIN ASSIGNMENTS (TOP VIEWS)

4-BIT BISTABLE LATCHES

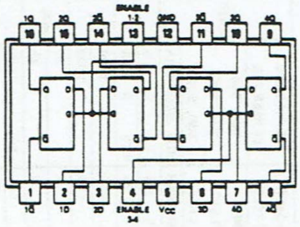
75

FUNCTION TABLE
(Each Latch)

INPUTS		OUTPUTS	
D	G	Q	$\bar{Q}$
L	H	L	H
H	H	H	L
X	L	Q_0	$\bar{Q}_0$

H = high level, L = low level, X = irrelevant
Q_0 = the level of Q before the high-to-low transistion of G

See page 7-35



SN5475 (J, W)
SN54L75 (J)
SN54LS75 (J, W)

SN7475 (J, N)
SN74L75 (J, N)
SN74LS75 (J, N)

DUAL J-K FLIP-FLOPS WITH PRESET AND CLEAR

76

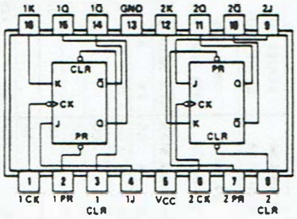
'76, 'H76
FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	⎍	L	L	Q_0	$\bar{Q}_0$
H	H	⎍	H	L	H	L
H	H	⎍	L	H	L	H
H	H	⎍	H	H	TOGGLE	

'LS76A
FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↓	L	L	Q_0	$\bar{Q}_0$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\bar{Q}_0$

See pages 6-46, 6-50, and 6-56



SN5476 (J, W)
SN54H76 (J, W)
SN54LS76A (J, W)

SN7476 (J, N)
SN74H76 (J, N)
SN74LS76A (J, N)

4-BIT BISTABLE LATCHES

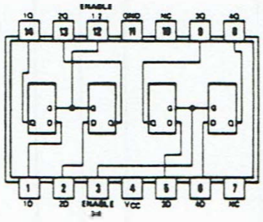
77

FUNCTION TABLE
(Each Latch)

INPUTS		OUTPUTS	
D	G	Q	$\bar{Q}$
L	H	L	H
H	H	H	L
X	L	Q_0	$\bar{Q}_0$

H = high level, L = low level, X = irrelevant
Q_0 = the level of Q before the high-to-low transistion of G

See page 7-35



SN5477 (W)
SN54L77 (T)
SN54LS77 (W)

See explanation of function tables on page 3-8.

*This configuration is nonstable; that is, it will not persist when preset and clear inputs return to their inactive (high) level.

POSITIVE-NAND GATES AND INVERTERS WITH TOTEM-POLE OUTPUTS

recommended operating conditions

	54 FAMILY 74 FAMILY	SERIES 54 SERIES 74 '00, '04, '10, '20, '30			SERIES 54H SERIES 74H 'H00, 'H04, 'H10, 'H20, 'H30			SERIES 54L SERIES 74L 'L00, 'L04, 'L10, 'L20, 'L30			SERIES 54LS SERIES 74LS 'LS00, 'LS04, 'LS10, 'LS20, 'LS30			SERIES 54S SERIES 74S 'S00, 'S04, 'S10, 'S20, 'S30, 'S133			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	54 Family	4.5	5	5.5	4.5	5	5.5	4.5	5	5.5	4.5	5	5.5	4.5	5	5.5	V
	74 Family	4.75	5	5.25	4.75	5	5.25	4.75	5	5.25	4.75	5	5.25	4.75	5	5.25	
High-level output current, I_{OH}	54 Family			−400			−500			−100			−400			−1000	μA
	74 Family			−400			−500			−200			−400			−1000	
Low-level output current, I_{OL}	54 Family			16			20			2			4			20	mA
	74 Family			16			20			3.6			8			20	
Operating free-air temperature, T_A	54 Family	−55		125	−55		125	−55		125	−55		125	−55		125	°C
	74 Family	0		70	0		70	0		70	0		70	0		70	

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST FIGURE	TEST CONDITIONS†			SERIES 54 SERIES 74 '00, '04, '10, '20, '30			SERIES 54H SERIES 74H 'H00, 'H04, 'H10, 'H20, 'H30			SERIES 54L SERIES 74L 'L00, 'L04, 'L10, 'L20, 'L30			SERIES 54LS SERIES 74LS 'LS00, 'LS04, 'LS10, 'LS20, 'LS30			SERIES 54S SERIES 74S 'S00, 'S04, 'S10, 'S20, 'S30, 'S133			UNIT
						MIN	TYP‡	MAX	MIN	TYP‡	MAX	MIN	TYP‡	MAX	MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage	1, 2				2			2			2			2			2			V
V_{IL}	Low-level input voltage	1, 2			54 Family			0.8			0.8			0.7			0.7			0.8	V
					74 Family			0.8			0.8			0.7			0.8			0.8	
V_{IK}	Input clamp voltage	3	V_{CC} = MIN, I_I = §					−1.5			−1.5						−1.5			−1.2	V
V_{OH}	High-level output voltage	1	V_{CC} = MIN, V_{IL} = V_{IL} max, I_{OH} = MAX		54 Family	2.4	3.4		2.4	3.5		2.4	3.3		2.5	3.4		2.5	3.4		V
					74 Family	2.4	3.4		2.4	3.5		2.4	3.2		2.7	3.4		2.7	3.4		
V_{OL}	Low-level output voltage	2	V_{CC} = MIN, V_{IH} = 2 V	I_{OL} = MAX	54 Family		0.2	0.4		0.2	0.4		0.15	0.3		0.25	0.4			0.5	V
					74 Family		0.2	0.4		0.2	0.4		0.2	0.4		0.25	0.5			0.5	
				I_{OL} = 4 mA	Series 74LS												0.4				
I_I	Input current at maximum input voltage	4	V_{CC} = MAX		V_I = 5.5 V			1			1			0.1						1	mA
					V_I = 7 V												0.1				
I_{IH}	High-level input current	4	V_{CC} = MAX		V_{IH} = 2.4 V			40			50			10							μA
					V_{IH} = 2.7 V												20			50	
I_{IL}	Low-level input current	5	V_{CC} = MAX		V_{IL} = 0.3 V									−0.18							mA
					V_{IL} = 0.4 V			1.6			−2						−0.4				
					V_{IL} = 0.5 V															−2	
I_{OS}	Short-circuit output current●	6	V_{CC} = MAX		54 Family	−20		−55	−40		−100	−3		−15	−20		−100	−40		−100	mA
					74 Family	−18		−55	−40		−100	−3		−15	−20		−100	−40		−100	
I_{CC}	Supply current	7	V_{CC} = MAX			See table on next page															mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.

‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.

§ I_I = −12 mA for SN54'/SN74', −8 mA for SN54H'/SN74H', and −18 mA for SN54LS'/SN74LS' and SN54S'/SN74S'.

● Not more than one output should be shorted at a time, and for SN54H'/SN74H', SN54LS'/SN74LS', and SN54S'/SN74S', duration of short-circuit should not exceed 1 second.

supply current¶

TYPE	I_{CCH} (mA) Total with outputs high		I_{CCL} (mA) Total with outputs low		I_{CC} (mA) Average per gate (50% duty cycle)
	TYP	MAX	TYP	MAX	TYP
'00	4	8	12	22	2
'04	6	12	18	33	2
'10	3	6	9	16.5	2
'20	2	4	6	11	2
'30	1	2	3	6	2
'H00	10	16.8	26	40	4.5
'H04	16	26	40	58	4.5
'H10	7.5	12.6	19.5	30	4.5
'H20	5	8.4	13	20	4.5
'H30	2.5	4.2	6.5	10	4.5
'L00	0.44	0.8	1.16	2.04	0.20
'L04	0.66	1.2	1.74	3.06	0.20
'L10	0.33	0.6	0.87	1.53	0.20
'L20	0.22	0.4	0.58	1.02	0.20
SN54L30	0.11	0.33	0.29	0.51	0.20
SN74L30	0.11	0.2	0.29	0.51	0.20
'LS00	0.8	1.6	2.4	4.4	0.4
'LS04	1.2	2.4	3.6	6.6	0.4
'LS10	0.6	-1.2	1.8	3.3	0.4
'LS20	0.4	0.8	1.2	2.2	0.4
'LS30	0.35	0.5	0.6	1.1	0.48
'S00	10	16	20	36	3.75
'S04	15	24	30	54	3.75
'S10	7.5	12	15	27	3.75
'S20	5	8	10	18	3.75
'S30	3	5	5.5	10	4.25
'S133	3	5	5.5	10	4.25

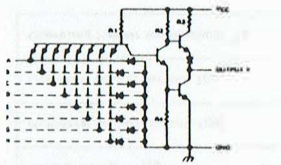
¶ Maximum values of I_{CC} are over the recommended operating ranges of V_{CC} and T_A; typical values are at V_{CC} = 5 V, T_A = 25°C.

switching characteristics at V_{CC} = 5 V, T_A = 25°C

TYPE	TEST CONDITIONS#	t_{PLH} (ns) Propagation delay time, low-to-high-level output			t_{PHL} (ns) Propagation delay time, high-to-low-level output		
		MIN	TYP	MAX	MIN	TYP	MAX
'00, '10	C_L = 15 pF, R_L = 400 Ω		11	22		7	15
'04, '20			12	22		8	15
'30			13	22		8	15
'H00	C_L = 25pF, R_L = 280 Ω		5.9	10		6.2	10
'H04			6	10		6.5	10
'H10			5.9	10		6.3	10
'H20			6	10		7	10
'H30			6.8	10		8.9	12
'L00, 'L04, 'L10, L20	C_L = 50 pF, R_L = 4 kΩ		35	60		31	60
'L30			35	60		70	100
'LS00, 'LS04 'LS10, 'LS20	C_L = 15 pF, R_L = 2 kΩ		9	15		10	15
'LS30			8	15		13	20
'S00, 'S04	C_L = 15 pF, R_L = 280 Ω		3	4.5		3	5
'S10, 'S20	C_L = 50 pF, R_L = 280 Ω		4.5			5	
'S30, 'S133	C_L = 15pF, R_L = 280 Ω		4	6		4.5	7
	C_L = 50 pF, R_L = 280 Ω		5.5			6.5	

#Load circuits and voltage waveforms are shown on pages 3-10 and 3-11.

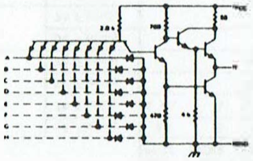
schematics (each gate)



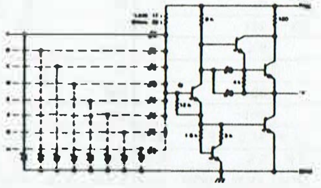
CIRCUIT	R1	R2	R3	R4
'00, '04, '10, '20, '30	4 k	1.6 k	130	1 k
'L00, 'L04, 'L10, 'L20, 'L30	40 k	20 k	500	12 k

'00, '04, '10, '20, '30
'L00, 'L04, 'L10, 'L20, 'L30, CIRCUITS

Input clamp diodes not on SN54L'/SN74L' circuits.

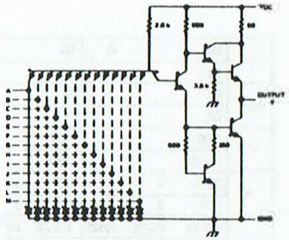


'H00, 'H04, 'H10, 'H20, 'H30 CIRCUITS



'LS00, 'LS04, 'LS10, 'LS20,
'LS30 CIRCUITS

*The 12-kΩ resistor is not on 'LS30.



'S00, 'S04, 'S10, 'S20,
'S30, 'S133 CIRCUITS

Resistor values shown are nominal and in ohms.

POSITIVE-NOR GATES WITH TOTEM-POLE OUTPUTS

recommended operating conditions

	54 FAMILY 74 FAMILY	SERIES 54 SERIES 74						SERIES 54L SERIES 74L			SERIES 54LS SERIES 74LS			SERIES 54S SERIES 74S			UNIT
		'02			'25, '27			'L02			'LS02, 'LS27			'S02, 'S260			
		MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	54 Family	4.5	5	5.5	4.5	5	5.5	4.5	5	5.5	4.5	5	5.5	4.5	5	5.5	V
	74 Family	4.75	5	5.25	4.75	5	5.25	4.75	5	5.25	4.75	5	5.25	4.75	5	5.25	
High-level output current, I_{OH}	54 Family			−400			−800			−100			−400			−1000	μA
	74 Family			−400			−800			−200			−400			−1000	
Low-level output current, I_{OL}	54 Family			16			16			2			4			20	mA
	74 Family			16			16			3.6			8			20	
Operating free-air temperature, T_A	54 Family	−55		125	−55		125	−55		125	−55		125	−55		125	°C
	74 Family	0		70	0		70	0		70	0		70	0		70	

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST FIGURE	TEST CONDITIONS†			SERIES 54 SERIES 74			SERIES 54L SERIES 74L			SERIES 54LS SERIES 74LS			SERIES 54S SERIES 74S			UNIT
							'02, '25, '27			'L02			'LS02 'LS27			'S02, 'S260			
							MIN	TYP‡	MAX	MIN	TYP‡	MAX	MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage		1, 2				2			2			2			2			V
V_{IL}	Low-level input voltage		1, 2			54 Family			0.8			0.7			0.7			0.8	V
						74 Family			0.8			0.7			0.8			0.8	
V_{IK}	Input clamp voltage		3	V_{CC} = MIN, I_I = §					−1.5						−1.5			−1.2	V
V_{OH}	High-level output voltage		1	V_{CC} = MIN, V_{IL} = V_{IL} max, I_{OH} = MAX		54 Family	2.4	3.4		2.4	3.3		2.5	3.4		2.5	3.4		V
						74 Family	2.4	3.4		2.4	3.2		2.7	3.4		2.7	3.4		
V_{OL}	Low-level output voltage		2	V_{CC} = MIN, V_{IH} = 2 V	I_{OL} = MAX	54 Family		0.2	0.4		0.15	0.3		0.25	0.4			0.5	V
						74 Family		0.2	0.4		0.2	0.4		0.35	0.5			0.5	
					I_{OL} = 4 mA	Series 74LS								0.25	0.4				
I_I	Input current at maximum input voltage		4	V_{CC} = MAX		V_I = 5.5 V			1			0.1						1	mA
						V_I = 7 V									0.1				
I_{IH}	High-level input current	Data inputs	4	V_{CC} = MAX		V_{IH} = 2.4 V			40			10							μA
		Strobe of '25							160										
		All inputs				V_{IH} = 2.7 V									20			50	
I_{IL}	Low-level input current	All inputs	5	V_{CC} = MAX		V_{IL} = 0.3 V						−0.18							mA
		Data inputs				V_{IL} = 0.4 V			−1.6						−0.4				
		Strobe of '25							−6.4										
		All inputs				V_{IL} = 0.5 V												−2	
I_{OS}	Short-circuit output current¶		6	V_{CC} = MAX		54 Family	−20		−55	−3		−15	−20		−100	−40		−100	mA
						74 Family	−18		−55	−3		−15	−20		−100	−40		−100	
I_{CC}	Supply current		7	V_{CC} = MAX			See table on next page												mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.

‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.

§ I_I = −12 mA for SN54'/SN74' and −18 mA for SN54LS'/SN74LS' and SN54S/SN74S'.

¶ Not more than one output should be shorted at a time, and for SN54LS'/SN74LS' and SN54S'/SN74S', duration of output short-circuit should not exceed one second.

supply current¶

TYPE	I_{CCH} (mA) Total with outputs high		I_{CCL} (mA) Total with outputs low		I_{CC} (mA) Average per gate (50% duty cycle)
	TYP	MAX	TYP	MAX	TYP
'02	8	16	14	27	2.75
'25	8	16	10	19	2.25
'27	10	16	16	26	4.34
'L02	0.8	1.6	1.4	2.6	0.275
'LS02	1.6	3.2	2.8	5.4	0.55
'LS27	2.0	4	3.4	6.8	0.9
'S02	17	29	26	45	5.38
'S260	17	29	26	45	10.75

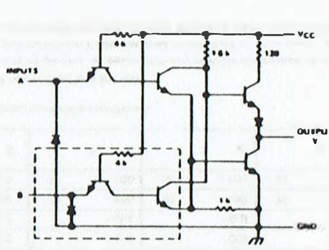
¶Maximum values of I_{CC} are over the recommended operating ranges of V_{CC} and T_A, typical values are at $V_{CC} = 5$ V, $T_A = 25°C$

switching characteristics at $V_{CC} = 5$ V, $T_A = 25°C$

TYPE	TEST CONDITIONS#	t_{PLH} (ns) Propagation delay time, low-to-high-level output			t_{PHL} (ns) Propagation delay time, high-to-low-level output		
		MIN	TYP	MAX	MIN	TYP	MAX
'02	$C_L = 15$ pF, $R_L = 400\ \Omega$		12	22		8	15
'25			13	22		8	15
'27			10	15		7	11
'L02	$C_L = 50$ pF, $R_L = 4\ k\Omega$		31	60		35	60
'LS02, 'LS27	$C_L = 15$ pF, $R_L = 2\ k\Omega$		10	15		10	15
'S02	$C_L = 15$ pF, $R_L = 280\ \Omega$		3.5	5.5		3.5	5.5
	$C_L = 50$ pF, $R_L = 280\ \Omega$		5			5	
'S260	$C_L = 15$ pF, $R_L = 280\ \Omega$		4	5.5		4	6

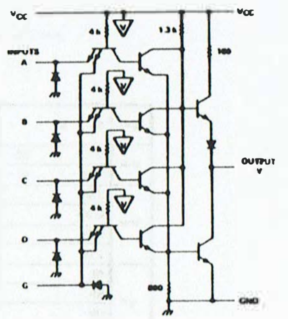
#Load circuit and voltage waveforms are shown on pages 3-10 and 3-11.

schematics (each gate)



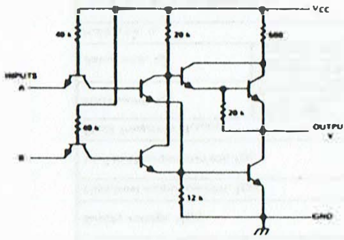
The portion of the schematic within the dashed lines is repeated for the C input of the '27.

'02, '27 CIRCUITS



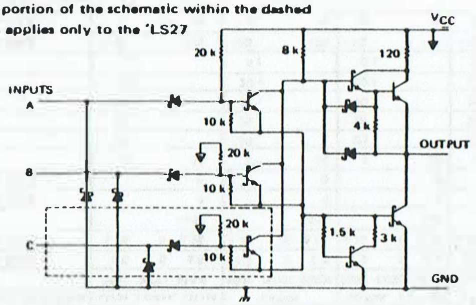
'25 CIRCUITS

Resistor values are nominal and in ohms.

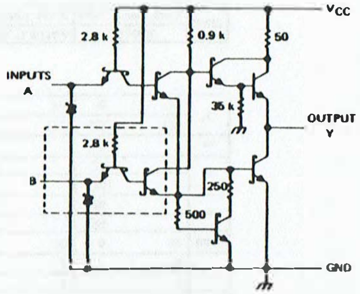


'L02 CIRCUITS

The portion of the schematic within the dashed lines applies only to the 'LS27



'LS02, 'LS27 CIRCUITS



The portion of the schematic within the dashed lines is repeated for each additional input of the 'S260, and the 0.9-kΩ resistor is changed to 0.6 kΩ.

'S02, 'S260 CIRCUITS

SERIES 54LS/74LS FLIP-FLOPS

recommended operating conditions

		SERIES 54LS/74LS	'LS73A,'LS107A,'LS113A MIN	NOM	MAX	'LS74A MIN	NOM	MAX	'LS76A, 'LS112A MIN	NOM	MAX	'LS78A, 'LS114A MIN	NOM	MAX	'LS109A MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}		Series 54LS	4.5	5	5.5	4.5	5	5.5	4.5	5	5.5	4.5	5	5.5	4.5	5	5.5	V
		Series 74LS	4.75	5	5.25	4.75	5	5.25	4.75	5	5.25	4.75	5	5.25	4.75	5	5.25	
High-level output current, I_{OH}					−400			−400			−400			−400			−400	μA
Low-level output current, I_{OL}		Series 54LS			4			4			4			4			4	mA
		Series 74LS			8			8			8			8			8	
Clock frequency, f_{clock}			0		30	0		25	0		30	0		30	0		25	MHz
Pulse width, t_w	Clock high		20			25			20			20			25			ns
	Preset or clear low		25			25			25			25			25			
Setup time, t_{su}	High-level data		20↓			20↑			20↓			20↓			35↑			ns
	Low-level data		20↓			20↑			20↓			20↓			25↑			
Hold time, t_h			0↓			5↑			0↓			0↓			5↑			ns
Operating free-air temperature, T_A		Series 54LS	−55		125	−55		125	−55		125	−55		125	−55		125	°C
		Series 74LS	0		70	0		70	0		70	0		70	0		70	

↑↓The arrow indicates the edge of the clock pulse used for reference: ↑ for the rising edge, ↓ for the falling edge.

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS†		'LS73A,'LS107A,'LS113A MIN	TYP‡	MAX	'LS74A MIN	TYP‡	MAX	'LS76A, 'LS112A MIN	TYP‡	MAX	'LS78A, 'LS114A MIN	TYP‡	MAX	'LS109A MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage				2			2			2			2			2			V
V_{IL}	Low-level input voltage	Series 54LS					0.7			0.7			0.7			0.7			0.7	V
		Series 74LS					0.8			0.8			0.8			0.8			0.8	
V_{IK}	Input clamp voltage		V_{CC} = MIN,	I_I = −18 mA			−1.5			−1.5			−1.5			−1.5			−1.5	V
V_{OH}	High-level output voltage	Series 54LS	V_{CC} = MIN, V_{IL} = V_{IL} max,	V_{IH} = 2 V, I_{OH} = −400 μA	2.5	3.4		2.5	3.4		2.5	3.4		2.5	3.4		2.5	3.4		V
		Series 74LS			2.7	3.4		2.7	3.4		2.7	3.4		2.7	3.4		2.7	3.4		
V_{OL}	Low-level output voltage	Series 54LS	V_{CC} = MIN, V_{IL} = V_{IL} max, V_{IH} = 2 V	I_{OL} = MAX		0.25	0.4		0.25	0.4		0.25	0.4		0.25	0.4		0.25	0.4	V
		Series 74LS				0.35	0.5		0.35	0.5		0.35	0.5		0.35	0.5		0.35	0.5	
		Series 74LS		I_{OL} = 4 mA		0.25	0.4		0.25	0.4		0.25	0.4		0.25	0.4		0.25	0.4	
I_I	Input current at maximum input voltage	D, J, K, or $\overline{K}$	V_{CC} = MAX,	V_I = 7 V			0.1			0.1			0.1			0.1			0.1	mA
		Clear					0.3			0.2			0.3			0.6			0.2	
		Preset					0.3			0.2			0.3			0.3			0.2	
		Clock					0.4			0.1			0.4			0.8			0.1	
I_{IH}	High-level input current	D, J, K, or $\overline{K}$	V_{CC} = MAX,	V_I = 2.7 V			20			20			20			20			20	μA
		Clear					60			40			60			120			40	
		Preset					60			40			60			60			40	
		Clock					80			20			80			160			20	
I_{IL}	Low-level input current	D, J, K, or $\overline{K}$	V_{CC} = MAX,	V_I = 0.4 V			−0.4			−0.4			−0.4			−0.4			−0.4	mA
		Clear					−0.8			−0.8			−0.8			−1.6			−0.8	
		Preset					−0.8			−0.8			−0.8			−0.8			−0.8	
		Clock					−0.8			−0.4			−0.8			−1.6			−0.4	
I_{OS}	Short-circuit output current◆	Series 54LS	V_{CC} = MAX,	See Note 1	−20		−100	−20		−100	−20		−100	−20		−100	−20		−100	mA
		Series 74LS			−20		−100	−20		−100	−20		−100	−20		−100	−20		−100	
I_{CC}	Supply current (Total)		V_{CC} = MAX,	See Note 2		4	6		4	8		4	6		4	6		4	8	mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.

‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.

◆ Not more than one output should be shorted at a time, and duration of short circuit should not exceed one second.

NOTES: 1. For certain devices where state commutation can be caused by shorting an output to ground, an equivalent test may be performed with V_O = 2.25 V and 2.125 V for the 54 family and the 74 family, respectively, with the minimum and maximum limits reduced to one half of their stated values.

2. With all outputs open, I_{CC} is measured with the Q and $\overline{Q}$ outputs high in turn. At the time of measurement, the clock input is grounded.

switching characteristics, $V_{CC} = 5\ V$, $T_A = 25°C$

PARAMETER¶	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	'LS73A, 'LS76A, 'LS78A, 'LS107A, 'LS112A, 'LS113A, 'LS114A			'LS74A, 'LS109A			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
f_{max}			C_L = 15 pF, R_L = 2 kΩ, See Note 2	30	45		25	33		MHz
t_{PLH}	Clear, preset, or clock (as appropriate)	Q or $\overline{Q}$			15	20		13	25	ns
t_{PHL}					15	20		25	40	ns

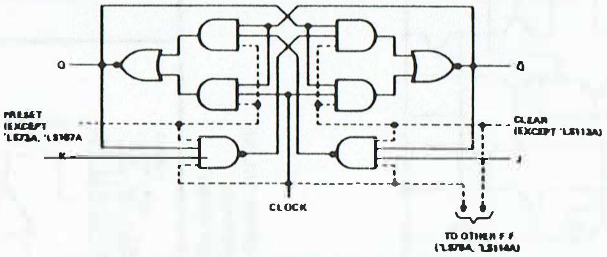
¶ f_{max} ≡ maximum clock frequency

t_{PLH} ≡ propagation delay time, low-to-high-level output

t_{PHL} ≡ propagation delay time, high-to-low-level output

NOTE 2: Load circuit and voltage waveforms are shown on page 3-11.

functional block diagrams and schematics of inputs and outputs


Q
Q̄
PRESET (EXCEPT 'LS73A, 'LS107A
K
CLEAR (EXCEPT 'LS113A)
J
CLOCK
TO OTHER F F ('LS78A, 'LS114A)

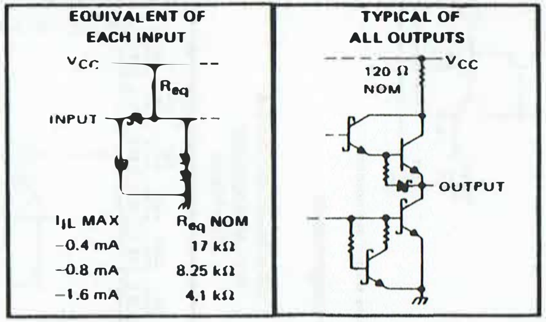
'LS73A, 'LS107A–DUAL J-K WITH CLEAR

'LS76A, 'LS112A–DUAL J-K WITH CLEAR AND PRESET

'LS78A, 'LS114A–DUAL J-K WITH PRESET, COMMON CLEAR, AND COMMON CLOCK

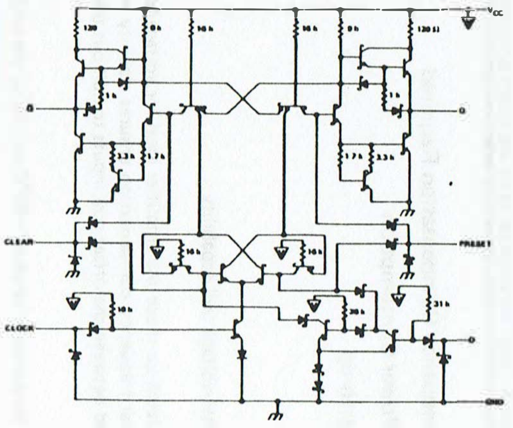
'LS113A–DUAL J-K WITH PRESET

'LS73A, 'LS76A, 'LS78A, 'LS107A, 'LS112A, 'LS113A, 'LS114A

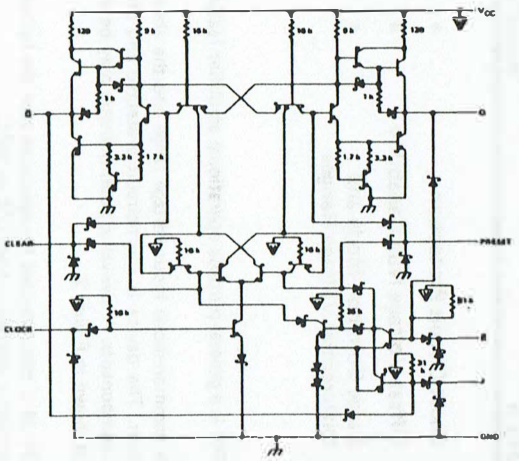


I_{IL} MAX	R_{eq} NOM
−0.4 mA	17 kΩ
−0.8 mA	8.25 kΩ
−1.6 mA	4.1 kΩ

schematics of 'LS74A and 'LS109A



'LS74A–DUAL D WITH CLEAR AND PRESET



'LS109A–DUAL J-K̄ WITH CLEAR AND PRESET

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

**LINEAR
INTEGRATED
CIRCUITS**

TYPES uA741M, uA741C
GENERAL-PURPOSE OPERATIONAL AMPLIFIERS

BULLETIN NO. DL-S 11363, NOVEMBER 1970–REVISED OCTOBER 1979

- Short-Circuit Protection
- Offset-Voltage Null Capability
- Large Common-Mode and Differential Voltage Ranges
- No Frequency Compensation Required
- Low Power Consumption
- No Latch-up

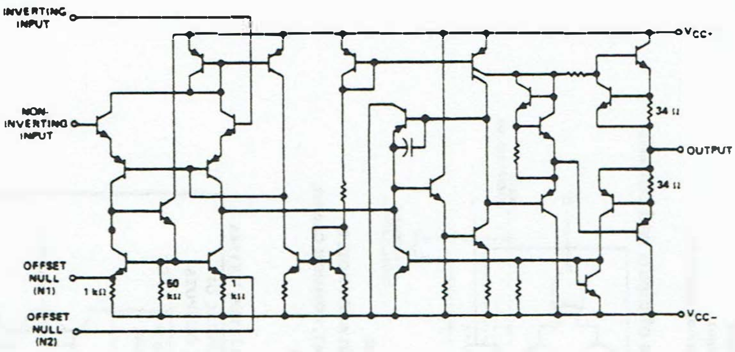
description

The uA741 is a general-purpose operational amplifier featuring offset-voltage null capability.

The high common-mode input voltage range and the absence of latch-up make the amplifier ideal for voltage-follower applications. The device is short-circuit protected and the internal frequency compensation ensures stability without external components. A low-value potentiometer may be connected between the offset null inputs to null out the offset voltage as shown in Figure 2.

The uA741M is characterized for operation over the full military temperature range of −55°C to 125°C; the uA741C is characterized for operation from 0°C to 70°C.

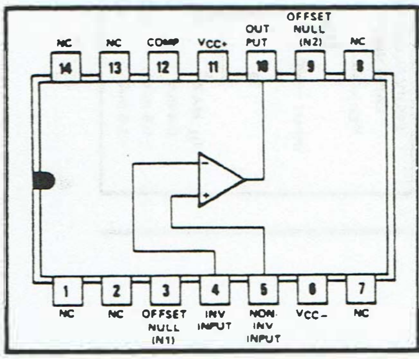
schematic



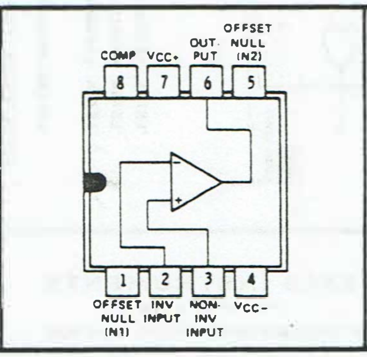
Resistor values shown are nominal

terminal assignments

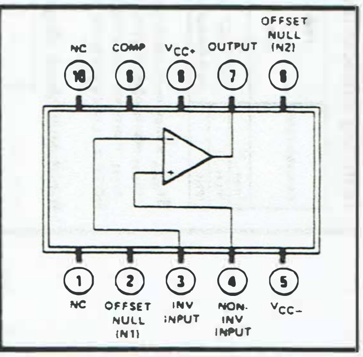
J OR N DUAL-IN-LINE OR
W FLAT PACKAGE
(TOP VIEW)



JG OR P DUAL-IN-LINE
PACKAGE
(TOP VIEW)



U
FLAT PACKAGE
(TOP VIEW)



NC—No internal connection

Copyright © 1979 by Texas Instruments Incorporated

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

TYPES uA741M, uA741C
GENERAL-PURPOSE OPERATIONAL AMPLIFIERS

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

		uA741M	uA741C	UNIT
Supply voltage V_{CC+} (see Note 1)		22	18	V
Supply voltage V_{CC-} (see Note 1)		−22	−18	V
Differential input voltage (see Note 2)		±30	±30	V
Input voltage (either input, see Notes 1 and 3)		±15	±15	V
Voltage between either offset null terminal (N1/N2) and V_{CC-}		±0.5	±0.5	V
Duration of output short-circuit (see Note 4)		unlimited	unlimited	
Continuous total power dissipation at (or below) 25°C free-air temperature (see Note 5)		500	500	mW
Operating free-air temperature range		−55 to 125	0 to 70	°C
Storage temperature range		−65 to 150	−65 to 150	°C
Lead temperature 1/16 inch (1,6 mm) from case for 60 seconds	J, JG, U, or W package	300	300	°C
Lead temperature 1/16 inch (1,6 mm) from case for 10 seconds	N or P package		260	°C

NOTES:
1. All voltage values, unless otherwise noted, are with respect to the midpoint between V_{CC+} and V_{CC-}.
2. Differential voltages are at the noninverting input terminal with respect to the inverting input terminal.
3. The magnitude of the input voltage must never exceed the magnitude of the supply voltage or 15 volts, whichever is less.
4. The output may be shorted to ground or either power supply. For the uA741M only, the unlimited duration of the short-circuit applies at (or below) 125°C case temperature or 75°C free-air temperature.
5. For operation above 25°C free-air temperature, refer to Dissipation Derating Curves, Section 2. In the J and JG packages, uA741M chips are alloy-mounted; uA741C chips are glass-mounted.

electrical characteristics at specified free-air temperature, $V_{CC+} = 15$ V, $V_{CC-} = -15$ V

PARAMETER		TEST CONDITIONS†		uA741M			uA741C			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
V_{IO}	Input offset voltage	$R_S \leq 10$ kΩ	25°C		1	5		1	6	mV
			Full range			6			7.5	
$\Delta V_{IO(adj)}$	Offset voltage adjust range		25°C		±15			±15		mV
I_{IO}	Input offset current		25°C		20	200		20	200	nA
			Full range			500			300	
I_{IB}	Input bias current		25°C		80	500		80	500	nA
			Full range			1500			800	
V_{ICR}	Common-mode input voltage range		25°C	±12	±13		±12	±13		V
			Full range	±12			±12			
V_{OPP}	Maximum peak-to-peak output voltage swing	$R_L = 10$ kΩ	25°C	24	28		24	28		V
		$R_L \geq 10$ kΩ	Full range	24			24			
		$R_L = 2$ kΩ	25°C	20	26		20	26		
		$R_L \geq 2$ kΩ	Full range	20			20			
A_{VD}	Large-signal differential voltage amplification	$R_L \geq 2$ kΩ, $V_O = \pm 10$ V	25°C	50	200		20	200		V/mV
			Full range	25			15			
r_i	Input resistance		25°C	0.3	2		0.3	2		MΩ
r_o	Output resistance	$V_O = 0$ V, See Note 6	25°C		75			75		Ω
C_i	Input capacitance		25°C		1.4			1.4		pF
CMRR	Common-mode rejection ratio	$R_S \leq 10$ kΩ	25°C	70	90		70	90		dB
			Full range	70			70			
k_{SVS}	Supply voltage sensitivity ($\Delta V_{IO}/\Delta V_{CC}$)	$R_S \leq 10$ kΩ	25°C		30	150		30	150	μV/V
			Full range			150			150	
I_{OS}	Short-circuit output current		25°C		±25	±40		±25	±40	mA
I_{CC}	Supply current	No load, No signal	25°C		1.7	2.8		1.7	2.8	mA
			Full range			3.3			3.3	
P_D	Total power dissipation	No load, No signal	25°C		50	85		50	85	mW
			Full range			100			100	

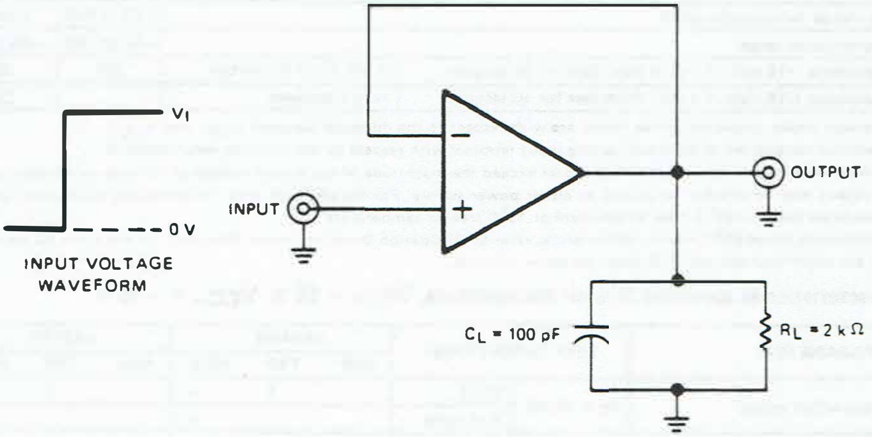
†All characteristics are specified under open-loop operation. Full range for uA741M is −55°C to 125°C and for uA741C is 0°C to 70°C.

NOTE 6: This typical value applies only at frequencies above a few hundred hertz because of the effects of drift and thermal feedback.

operating characteristics, $V_{CC+} = 15$ V, $V_{CC-} = -15$ V, $T_A = 25°C$

PARAMETER		TEST CONDITIONS	uA741M			uA741C			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_r	Rise time	V_I = 20 mV, R_L = 2 kΩ, C_L = 100 pF, See Figure 1		0.3			0.3		μs
	Overshoot factor			5%			5%		
SR	Slew rate at unity gain	V_I = 10 V, R_L = 2 kΩ, C_L = 100 pF, See Figure 1		0.5			0.5		V/μs

PARAMETER MEASUREMENT INFORMATION



INPUT VOLTAGE WAVEFORM

TEST CIRCUIT

FIGURE 1—RISE TIME, OVERSHOOT, AND SLEW RATE

TYPICAL APPLICATION DATA

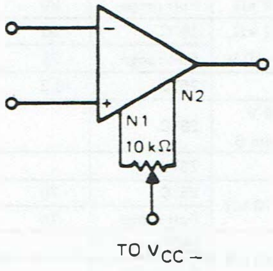


FIGURE 2—INPUT OFFSET VOLTAGE NULL CIRCUIT

TYPES uA741M, uA741C
GENERAL-PURPOSE OPERATIONAL AMPLIFIERS

TYPICAL CHARACTERISTICS

INPUT OFFSET CURRENT
vs
FREE-AIR TEMPERATURE

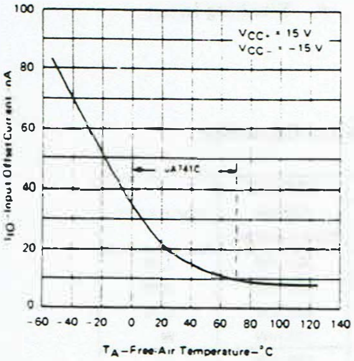
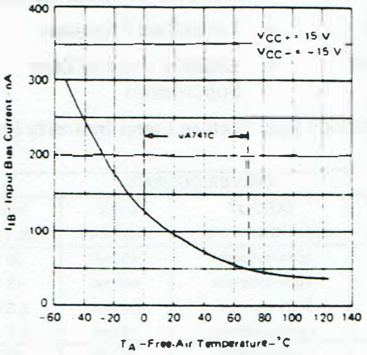


FIGURE 3

INPUT BIAS CURRENT
vs
FREE-AIR TEMPERATURE



FIGUER 4

MAXIMUM PEAK-TO-PEAK OUTPUT VOLTAGE
vs
LOAD RESISTANCE

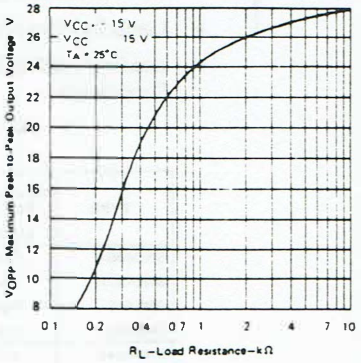


FIGURE 5

MAXIMUM PEAK-TO-PEAK OUTPUT VOLTAGE
vs
FREQUENCY

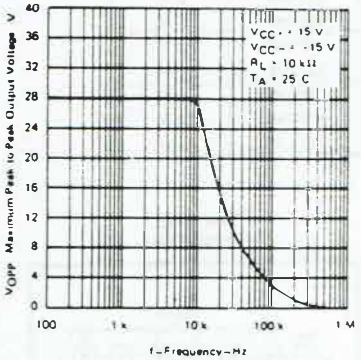


FIGURE 6

OPEN-LOOP LARGE-SIGNAL
DIFFERENTIAL
VOLTAGE AMPLIFICATION
vs
SUPPLY VOLTAGE

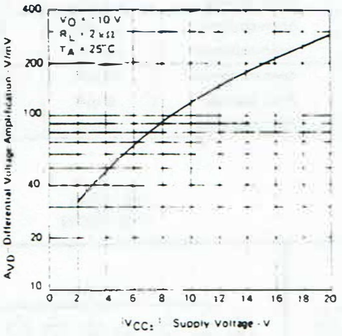


FIGURE 7

OPEN-LOOP LARGE-SIGNAL
DIFFERENTIAL
VOLTAGE AMPLIFICATION
vs
FREQUENCY

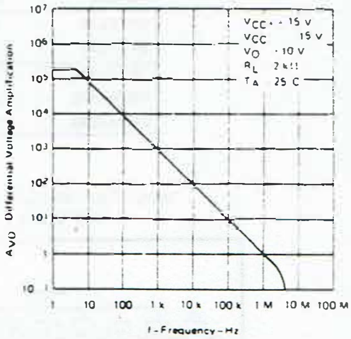


FIGURE 8

COMMON-MODE REJECTION RATIO
vs
FREQUENCY

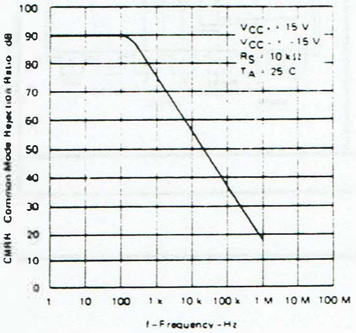


FIGURE 9

OUTPUT VOLTAGE
vs
ELAPSED TIME

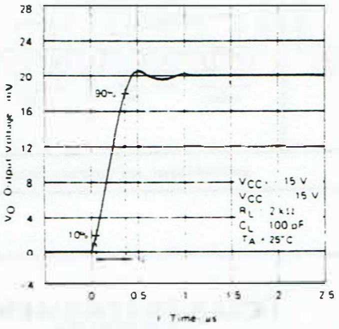


FIGURE 10

VOLTAGE-FOLLOWER
LARGE-SIGNAL PULSE RESPONSE

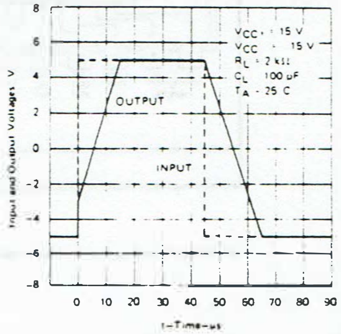


FIGURE 11

TYPES SN5446A, '47A, '48, '49, SN54L46, 'L47, SN54LS47, 'LS48, 'LS49, SN7446A, '47A, '48, SN74L46, 'L47, SN74LS47, 'LS48, 'LS49
BCD-TO-SEVEN-SEGMENT DECODERS/DRIVERS

BULLETIN NO. DL-S 7611811, MARCH 1974–REVISED OCTOBER 1976

'46A, '47A, 'L46, 'L47, 'LS47 feature

- Open-Collector Outputs Drive Indicators Directly
- Lamp-Test Provision
- Leading/Trailing Zero Suppression

'48, 'LS48 feature

- Internal Pull-Ups Eliminate Need for External Resistors
- Lamp-Test Provision
- Leading/Trailing Zero Suppression

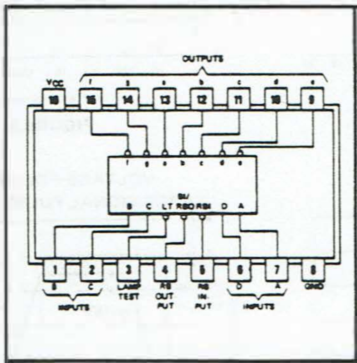
'49, 'LS49 feature

- Open-Collector Outputs
- Blanking Input

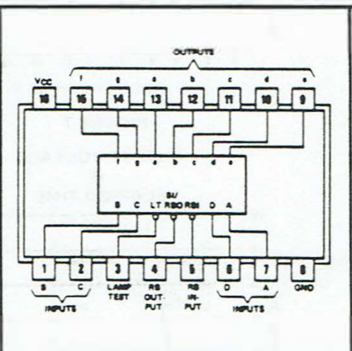
- All Circuit Types Feature Lamp Intensity Modulation Capability

TYPE	DRIVER OUTPUTS				TYPICAL POWER DISSIPATION	PACKAGES
	ACTIVE LEVEL	OUTPUT CONFIGURATION	SINK CURRENT	MAX VOLTAGE		
SN5446A	low	open-collector	40 mA	30 V	320 mW	J, W
SN5447A	low	open-collector	40 mA	15 V	320 mW	J, W
SN5448	high	2-kΩ pull-up	6.4 mA	5.5 V	265 mW	J, W
SN5449	high	open-collector	10 mA	5.5 V	165 mW	W
SN54L46	low	open-collector	20 mA	30 V	160 mW	J
SN54L47	low	open-collector	20 mA	15 V	160 mW	J
SN54LS47	low	open-collector	12 mA	15 V	35 mW	J, W
SN54LS48	high	2-kΩ pull-up	2 mA	5.5 V	125 mW	J, W
SN54LS49	high	open-collector	4 mA	5.5 V	40 mW	J, W
SN7446A	low	open-collector	40 mA	30 V	320 mW	J, N
SN7447A	low	open-collector	40 mA	15 V	320 mW	J, N
SN7448	high	2-kΩ pull-up	6.4 mA	5.5 V	265 mW	J, N
SN74L46	low	open-collector	20 mA	30 V	160 mW	J, N
SN74L47	low	open-collector	20 mA	15 V	160 mW	J, N
SN74LS47	low	open-collector	24 mA	15 V	35 mW	J, N
SN74LS48	high	2-kΩ pull-up	6 mA	5.5 V	125 mW	J, N
SN74LS49	high	open-collector	8 mA	5.5 V	40 mW	J, N

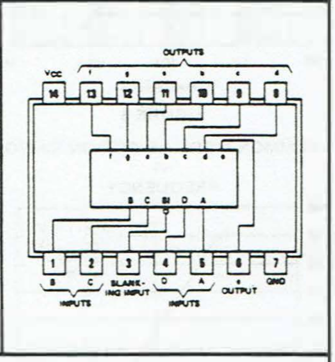
'46A, '47A, 'L46, 'L47, 'LS47
(TOP VIEW)



'48, 'LS48
(TOP VIEW)



'49, 'LS49
(TOP VIEW)



positive logic: see function tables

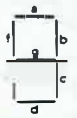
TYPES SN5446A, '47A, '48, '49, SN54L46, 'L47, SN54LS47, 'LS48, 'LS49, SN7446A, '47A, '48, SN74L46, 'L47, SN74LS47, 'LS48, 'LS49 BCD-TO-SEVEN-SEGMENT DECODERS/DRIVERS

description

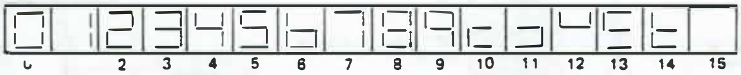
The '46A, 'L46, '47A, 'L47, and 'LS47 feature active-low outputs designed for driving common-anode VLEDs or incandescent indicators directly, and the '48, '49, 'LS48, 'LS49 feature active-high outputs for driving lamp buffers or common-cathode VLEDs. All of the circuits except '49 and 'LS49 have full ripple-blanking input/output controls and a lamp test input. The '49 and 'LS49 circuits incorporate a direct blanking input. Segment identification and resultant displays are shown below. Display patterns for BCD input counts above 9 are unique symbols to authenticate input conditions.

The '46A, '47A, '48, 'L46, 'L47, 'LS47, and 'LS48 circuits incorporate automatic leading and/or trailing-edge zero-blanking control (RBI and RBO). Lamp test (LT) of these types may be performed at any time when the BI/RBO node is at a high level. All types (including the '49 and 'LS49) contain an overriding blanking input (BI) which can be used to control the lamp intensity by pulsing or to inhibit the outputs. Inputs and outputs are entirely compatible for use with TTL or DTL logic outputs.

The SN54246/SN74246 through '249 and the SN54LS247/SN74LS247 through 'LS249 compose the 6 and the 9 with tails and have been designed to offer the designer a choice between two indicator fonts. The SN54249/SN74249 and SN54LS249/SN74LS249 are 16-pin versions of the 14-pin SN5449 and 'LS49. Included in the '249 circuit and 'LS249 circuits are the full functional capability for lamp test and ripple blanking, which is not available in the '49 or 'LS49 circuit.



SEGMENT IDENTIFICATION



NUMERICAL DESIGNATIONS AND RESULTANT DISPLAYS

'46A, '47A, 'L46, 'L47, 'LS47 FUNCTION TABLE

DECIMAL OR FUNCTION	INPUTS						BI/RBO†	OUTPUTS							NOTE
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g	
0	H	H	L	L	L	L	H	ON	ON	ON	ON	ON	ON	OFF	1
1	H	X	L	L	L	H	H	OFF	ON	ON	OFF	OFF	OFF	OFF	
2	H	X	L	L	H	L	H	ON	ON	OFF	ON	ON	OFF	ON	
3	H	X	L	L	H	H	H	ON	ON	ON	ON	OFF	OFF	ON	
4	H	X	L	H	L	L	H	OFF	ON	ON	OFF	OFF	ON	ON	
5	H	X	L	H	L	H	H	ON	OFF	ON	ON	OFF	ON	ON	
6	H	X	L	H	H	L	H	OFF	OFF	ON	ON	ON	ON	ON	
7	H	X	L	H	H	H	H	ON	ON	ON	OFF	OFF	OFF	OFF	
8	H	X	H	L	L	L	H	ON	ON	ON	ON	ON	ON	ON	
9	H	X	H	L	L	H	H	ON	ON	ON	OFF	OFF	ON	ON	
10	H	X	H	L	H	L	H	OFF	OFF	OFF	ON	ON	OFF	ON	
11	H	X	H	L	H	H	H	OFF	OFF	ON	ON	OFF	OFF	ON	
12	H	X	H	H	L	L	H	OFF	ON	OFF	OFF	OFF	ON	ON	
13	H	X	H	H	L	H	H	ON	OFF	OFF	ON	OFF	ON	ON	
14	H	X	H	H	H	L	H	OFF	OFF	OFF	ON	ON	ON	ON	
15	H	X	H	H	H	H	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF	
BI	X	X	X	X	X	X	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	2
RBI	H	L	L	L	L	L	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	3
LT	L	X	X	X	X	X	H	ON	ON	ON	ON	ON	ON	ON	4

H = high level, L = low level, X = irrelevant

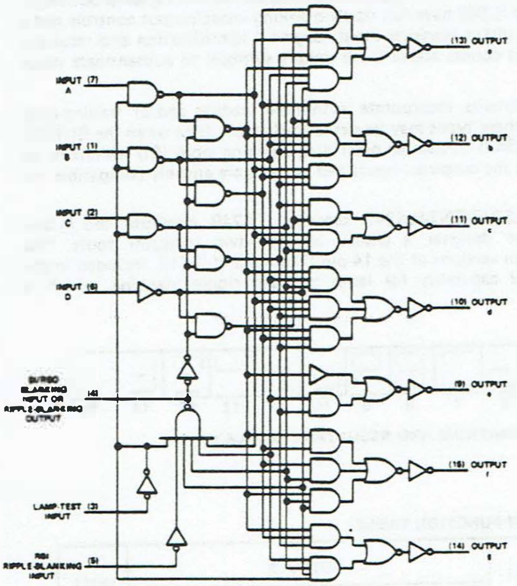
NOTES:
1. The blanking input (BI) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input (RBI) must be open or high if blanking of a decimal zero is not desired.
2. When a low logic level is applied directly to the blanking input (BI), all segment outputs are off regardless of the level of any other input.
3. When ripple-blanking input (RBI) and inputs A, B, C, and D are at a low level with the lamp test input high, all segment outputs go off and the ripple-blanking output (RBO) goes to a low level (response condition).
4. When the blanking input/ripple blanking output (BI/RBO) is open or held high and a low is applied to the lamp-test input, all segment outputs are on.

† BI/RBO is wire-AND logic serving as blanking input (BI) and/or ripple-blanking output (RBO).

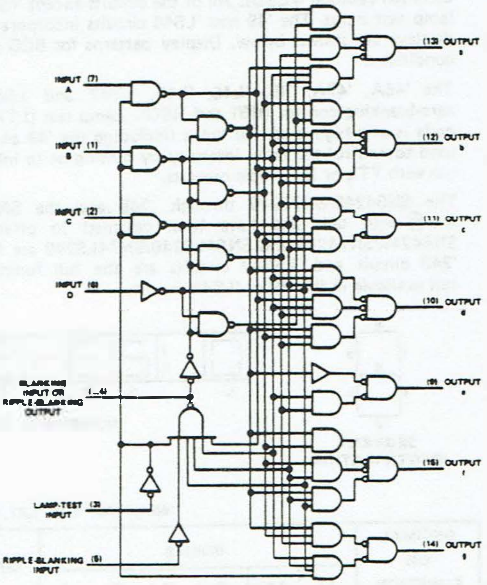
TYPES SN5446A, '47A, '48, '49, SN54L46, 'L47, SN54LS47, 'LS48, 'LS49, SN7446A, '47A, '48, SN74L46, 'L47, SN74LS47, 'LS48, 'LS49 BCD-TO-SEVEN-SEGMENT DECODERS/DRIVERS

functional block diagrams

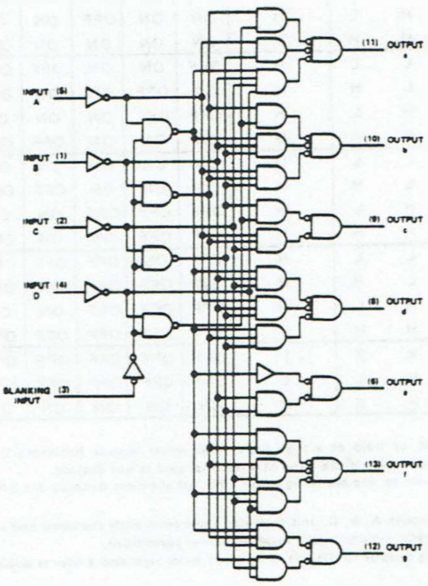
'46A, '47A, 'L46, 'L47, 'LS47



'48, 'LS48



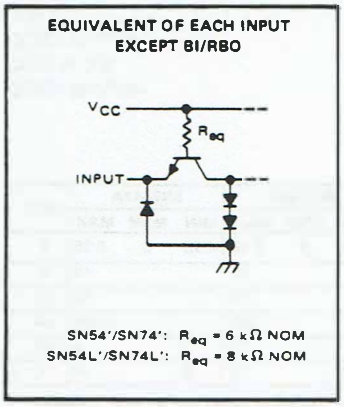
'49, 'LS49



TYPES SN5446A, '47A, '48, '49, SN54L46, 'L47, SN7446A, '47A, '48, SN74L46, 'L47 BCD-TO-SEVEN-SEGMENT DECODERS/DRIVERS

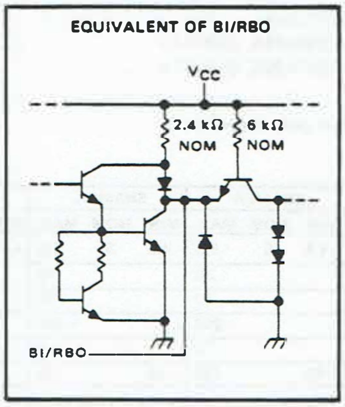
schematics of inputs and outputs

'46A, '47A, '48, '49, 'L46, 'L47

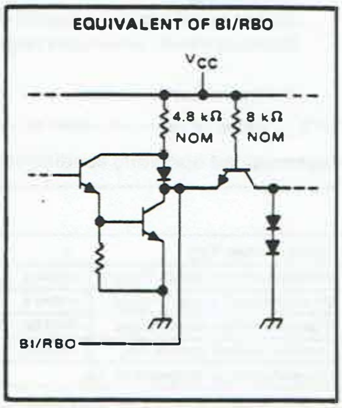

EQUIVALENT OF EACH INPUT EXCEPT BI/RBO

SN54'/SN74': R_{eq} = 6 kΩ NOM
SN54L'/SN74L': R_{eq} = 8 kΩ NOM

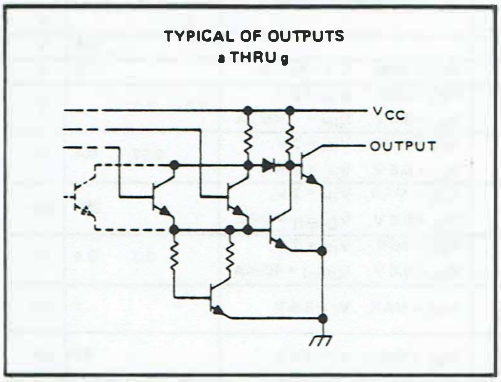
'46A, '47A, '48


EQUIVALENT OF BI/RBO

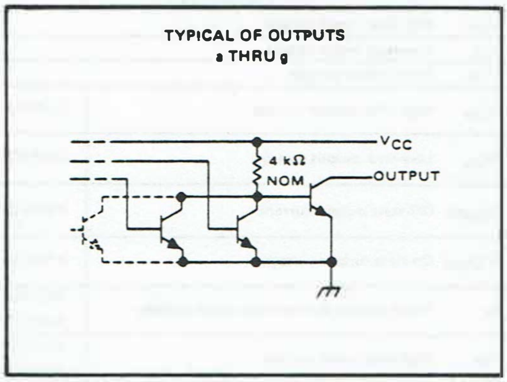
'L46, 'L47


EQUIVALENT OF BI/RBO

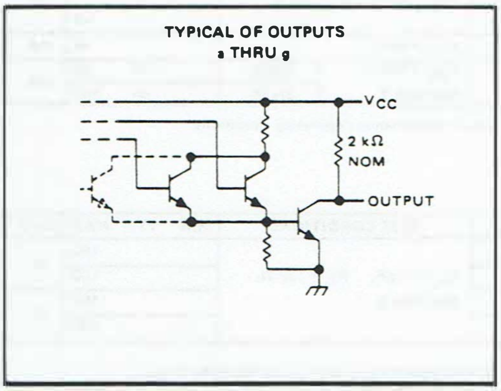
'46A, '47A


TYPICAL OF OUTPUTS a THRU g

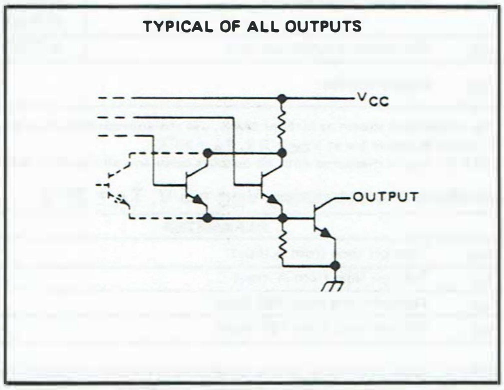
'L46, 'L47


TYPICAL OF OUTPUTS a THRU g

'48


TYPICAL OF OUTPUTS a THRU g

'49


TYPICAL OF ALL OUTPUTS

TYPES SN5446A, SN5447A, SN7446A, SN7447A BCD-TO-SEVEN-SEGMENT DECODERS/DRIVERS

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1)		7 V
Input voltage		5.5 V
Current forced into any output in the off state		1 mA
Operating free-air temperature range:	SN5446A, SN5447A	−55°C to 125°C
	SN7446A, SN7447A	0°C to 70°C
Storage temperature range		−65°C to 150°C

NOTE 1: Voltage values are with respect to network ground terminal.

recommended operating conditions

		SN5446A			SN5447A			SN7446A			SN7447A			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}		4.5	5	5.5	4.5	5	5.5	4.75	5	5.25	4.75	5	5.25	V
Off-state output voltage, $V_{O(off)}$	a thru g			30			15			30			15	V
On-state output current, $I_{O(on)}$	a thru g			40			40			40			40	mA
High-level output current, I_{OH}	BI/RBO			−200			−200			−200			−200	μA
Low-level output current, I_{OL}	BI/RBO			8			8			8			8	mA
Operating free-air temperature, T_A		−55		125	−55		125	0		70	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS†		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage				2			V
V_{IL}	Low-level input voltage						0.8	V
V_{IK}	Input clamp voltage		V_{CC} = MIN, I_I = −12 mA				−1.5	V
V_{OH}	High-level output voltage	BI/RBO	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8V, I_{OH} = −200 μA		2.4	3.7		V
V_{OL}	Low-level output voltage	BI/RBO	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8V, I_{OL} = 8 mA			0.27	0.4	V
$I_{O(off)}$	Off-state output current	a thru g	V_{CC} = MAX, V_{IH} = 2 V, V_{IL} = 0.8V, $V_{O(off)}$ = MAX				250	μA
$V_{O(on)}$	On-state output voltage	a thru g	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8V, $I_{O(on)}$ = 40 mA			0.3	0.4	V
I_I	Input current at maximum input voltage	Any input except BI/RBO	V_{CC} = MAX, V_I = 5.5 V				1	mA
I_{IH}	High-level input current	Any input except BI/RBO	V_{CC} = MAX, V_I = 2.4 V				40	μA
I_{IL}	Low-level input current	Any input except BI/RBO	V_{CC} = MAX, V_I = 0.4 V				−1.6	mA
		BI/RBO					−4	
I_{OS}	Short-circuit output current	BI/RBO	V_{CC} = MAX				−4	mA
I_{CC}	Supply current		V_{CC} = MAX, See Note 2	SN54'		64	85	mA
				SN74'		64	103	

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.

‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.

NOTE 2: I_{CC} is measured with all outputs open and all inputs at 4.5 V.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{off}	Turn-off time from A input	C_L = 15 pF, R_L = 120 Ω, See Note 3			100	ns
t_{on}	Turn-on time from A input				100	
t_{off}	Turn-off time from RBI input				100	ns
t_{on}	Turn-on time from RBI input				100	

NOTE 3: Load circuit and voltage waveforms are shown on page 3-10; t_{off} corresponds to t_{PLH} and t_{on} corresponds to t_{PHL}.

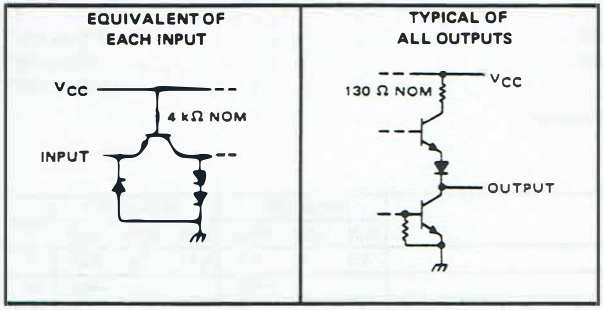
TTL MSI

TYPES SN5486, SN54L86, SN54LS86, SN54S86, SN7486, SN74L86, SN74LS86, SN74S86
QUADRUPLE 2-INPUT EXCLUSIVE-OR GATES

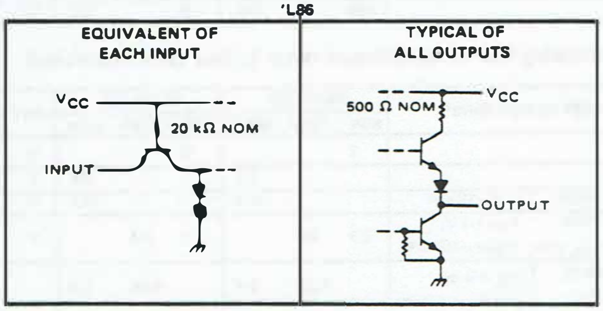
BULLETIN NO. DL-S 7611825, DECEMBER 1972–REVISED OCTOBER 1976

schematics of inputs and outputs

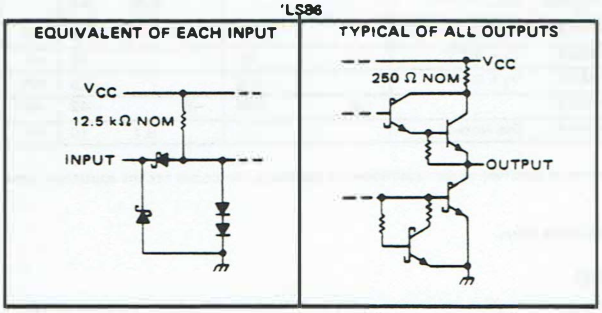
'86



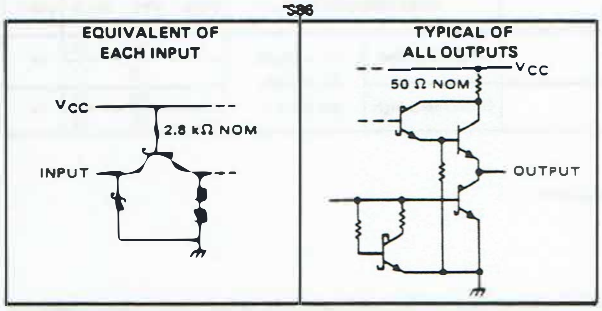
'L86



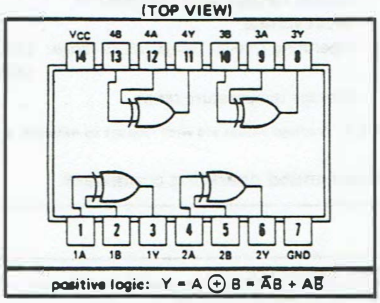
'LS86



'S86

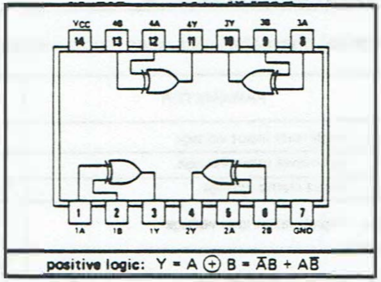


SN54', SN54LS', SN54S' . . . J OR W PACKAGE
SN74', SN74LS', SN74S' . . . J OR N PACKAGE
(TOP VIEW)



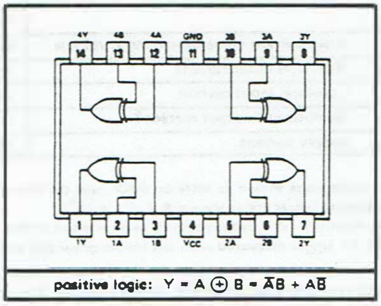
positive logic: $Y = A \oplus B = \bar{A}B + A\bar{B}$

SN54L86 . . . J PACKAGE
SN74L86 . . . J OR N PACKAGE
(TOP VIEW)



positive logic: $Y = A \oplus B = \bar{A}B + A\bar{B}$

SN54L86 . . . T PACKAGE (TOP VIEW)



positive logic: $Y = A \oplus B = \bar{A}B + A\bar{B}$

FUNCTION TABLE

INPUTS		OUTPUT
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	L

H = high level, L = low level

TYPE	TYPICAL AVERAGE PROPAGATION DELAY TIME	TYPICAL TOTAL POWER DISSIPATION
'86	14 ns	150 mW
'L86	55 ns	15 mW
'LS86	10 ns	30.5 mW
'S86	7 ns	250 mW

TYPES SN54LS86, SN74LS86
QUADRUPLE 2-INPUT EXCLUSIVE-OR GATES

REVISED OCTOBER 1976

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1)		7 V
Input voltage		7 V
Operating free-air temperature range:	SN54LS86	−55°C to 125°C
	SN74LS86	0°C to 70°C
Storage temperature range		−65°C to 150°C

NOTE 1: Voltage values are with respect to network ground terminal.

recommended operating conditions

	SN54LS86			SN74LS86			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS†		SN54LS86			SN74LS86			UNIT
				MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage			2			2			V
V_{IL}	Low-level input voltage					0.7			0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN,	$I_I = -18$ mA			−1.5			−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, $V_{IL} = V_{IL}$ max,	V_{IH} = 2 V, $I_{OH} = -400$ μA	2.5	3.4		2.7	3.4		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, $V_{IL} = V_{IL}$max	I_{OL} = 4 mA		0.25	0.4		0.25	0.4	V
			I_{OL} = 8 mA					0.35	0.5	
I_I	Input current at maximum input voltage	V_{CC} = MAX,	V_I = 7 V			0.2			0.2	mA
I_{IH}	High-level input current	V_{CC} = MAX,	V_I = 2.7 V			40			40	μA
I_{IL}	Low-level input current	V_{CC} = MAX,	V_I = 0.4 V			−0.8			−0.8	mA
I_{OS}	Short-circuit output current§	V_{CC} = MAX		−6		−40	−5		−42	mA
I_{CC}	Supply current	V_{CC} = MAX,	See Note 2		6.1	10		6.1	10	mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable type.

‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.

§ Not more than one output should be shorted at a time.

NOTE 2: I_{CC} is measured with the inputs grounded and the outputs open.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER¶	FROM (INPUT)	TEST CONDITIONS		MIN	TYP	MAX	UNIT
t_{PLH}	A or B	Other input low	C_L = 15 pF, R_L = 2 kΩ, See Note 7		12	23	ns
t_{PHL}					10	17	
t_{PLH}	A or B	Other input high			20	30	ns
t_{PHL}					13	22	

¶ t_{PLH} ≡ propagation delay time, low-to-high-level output

t_{PHL} ≡ propagation delay time, high-to-low-level output

NOTE 7: Load circuit and voltage waveforms are shown on page 3-11.

TTL
MSI

TYPES SN5490A, SN5492A, SN5493A, SN54L90, SN54L93, SN54LS90, SN54LS92, SN54LS93, SN7490A, SN7492A, SN7493A, SN74L90, SN74L93, SN74LS90, SN74LS92, SN74LS93 DECADE, DIVIDE-BY-TWELVE, AND BINARY COUNTERS

BULLETIN NO. DL-S 7611807, MARCH 1974–REVISED OCTOBER 1976

'90A, 'L90, 'LS90 . . . DECADE COUNTERS

'92A, 'LS92 . . . DIVIDE-BY-TWELVE COUNTERS

'93A, 'L93, 'LS93 . . . 4-BIT BINARY COUNTERS

TYPES	TYPICAL POWER DISSIPATION
'90A	145 mW
'L90	20 mW
'LS90	45 mW
'92A, '93A	130 mW
'LS92, 'LS93	45 mW
'L93	16 mW

description

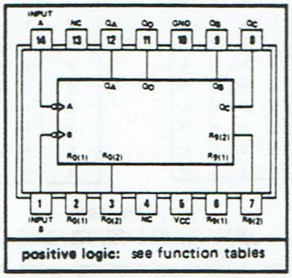
Each of these monolithic counters contains four master-slave flip-flops and additional gating to provide a divide-by-two counter and a three-stage binary counter for which the count cycle length is divide-by-five for the '90A, 'L90, and 'LS90, divide-by-six for the '92A and 'LS92, and divide-by-eight for the '93A, 'L93, and 'LS93.

All of these counters have a gated zero reset and the '90A, 'L90, and 'LS90 also have gated set-to-nine inputs for use in BCD nine's complement applications.

To use their maximum count length (decade, divide-by-twelve, or four-bit binary) of these counters, the B input is connected to the Q_A output. The input count pulses are applied to input A and the outputs are as described in the appropriate function table. A symmetrical divide-by-ten count can be obtained from the '90A, 'L90, or 'LS90 counters by connecting the Q_D output to the A input and applying the input count to the B input which gives a divide-by-ten square wave at output Q_A.

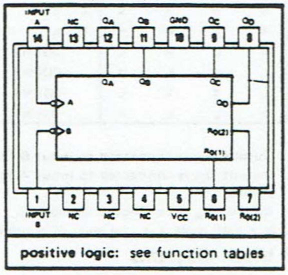
SN54', SN54LS' . . . J OR W PACKAGE
SN54L' . . . J OR T PACKAGE
SN54', SN74L', SN74LS' . . . J OR N PACKAGE

'90A, 'L90, 'LS90 (TOP VIEW)



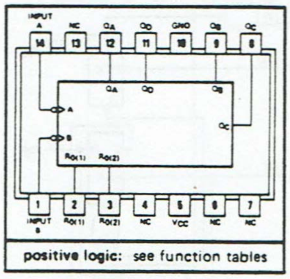
positive logic: see function tables

'92A, 'LS92, (TOP VIEW)



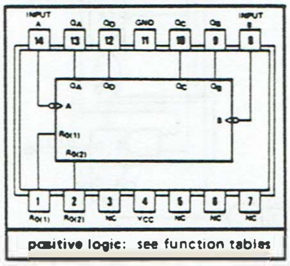
positive logic: see function tables

'93A, 'LS93 (TOP VIEW)



positive logic: see function tables

'L93 (TOP VIEW)



positive logic: see function tables

NC–No internal connection

TYPES SN5490A, '92A, '93A, SN54L90, 'L93, SN54LS90, 'LS92, 'LS93, SN7490A, '92A, '93A, SN74L90, 'L93, SN74LS90, 'LS92, 'LS93 DECADE, DIVIDE-BY-TWELVE, AND BINARY COUNTERS

'90A, 'L90, 'LS90
BCD COUNT SEQUENCE
(See Note A)

COUNT	OUTPUT			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H

'90A, 'L90, 'LS90
BI-QUINARY (5-2)
(See Note B)

COUNT	OUTPUT			
	Q_A	Q_D	Q_C	Q_B
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	H	L	L	L
6	H	L	L	H
7	H	L	H	L
8	H	L	H	H
9	H	H	L	L

'92A, 'LS92
COUNT SEQUENCE
(See Note C)

COUNT	OUTPUT			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	H	L	L	L
7	H	L	L	H
8	H	L	H	L
9	H	L	H	H
10	H	H	L	L
11	H	H	L	H

'93A, 'L93, 'LS93
COUNT SEQUENCE
(See Note C)

COUNT	OUTPUT			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H
10	H	L	H	L
11	H	L	H	H
12	H	H	L	L
13	H	H	L	H
14	H	H	H	L
15	H	H	H	H

'90A, 'L90, 'LS90
RESET/COUNT FUNCTION TABLE

RESET INPUTS				OUTPUT			
$R_{0(1)}$	$R_{0(2)}$	$R_{9(1)}$	$R_{9(2)}$	Q_D	Q_C	Q_B	Q_A
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
X	L	X	L	COUNT			
L	X	L	X	COUNT			
L	X	X	L	COUNT			
X	L	L	X	COUNT			

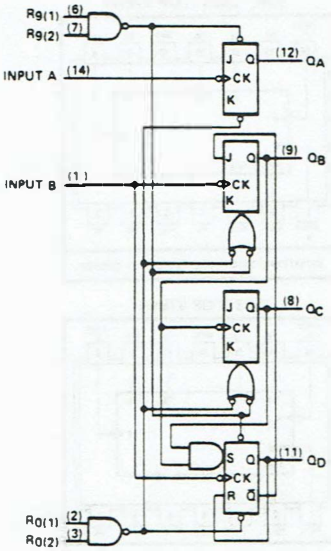
'92A, 'LS92, '93A, 'L93, 'LS93
RESET/COUNT FUNCTION TABLE

RESET INPUTS		OUTPUT			
$R_{0(1)}$	$R_{0(2)}$	Q_D	Q_C	Q_B	Q_A
H	H	L	L	L	L
L	X	COUNT			
X	L	COUNT			

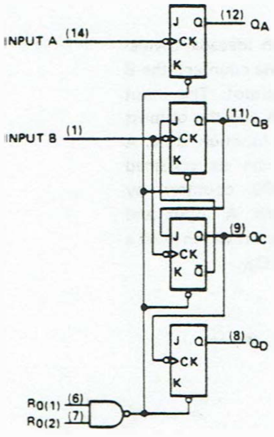
NOTES:
A. Output Q_A is connected to input B for BCD count.
B. Output Q_D is connected to input A for bi-quinary count.
C. Output Q_A is connected to input B.
D. H = high level, L = low level, X = irrelevant

functional block diagrams

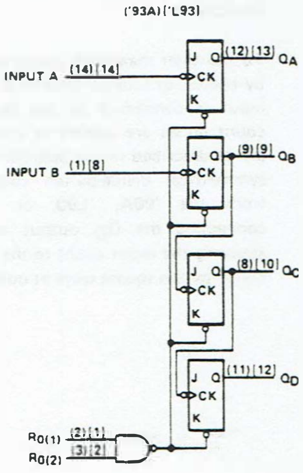
'90A, 'L90, 'LS90



'92A, 'LS92



'93A, 'L93, 'LS93



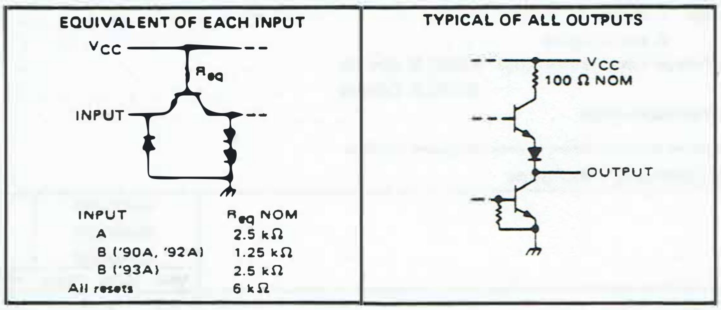
The J and K inputs shown without connection are for reference only and are functionally at a high level.

TYPES SN5490A, '92A, '93A, SN54L90, 'L93, SN54LS90, 'LS92, 'LS93, SN7490A, '92A, '93A, SN74L90, 'L93, SN74LS90, 'LS92, 'LS93 DECADE, DIVIDE-BY-TWELVE, AND BINARY COUNTERS

REVISED AUGUST 1977

schematics of inputs and outputs

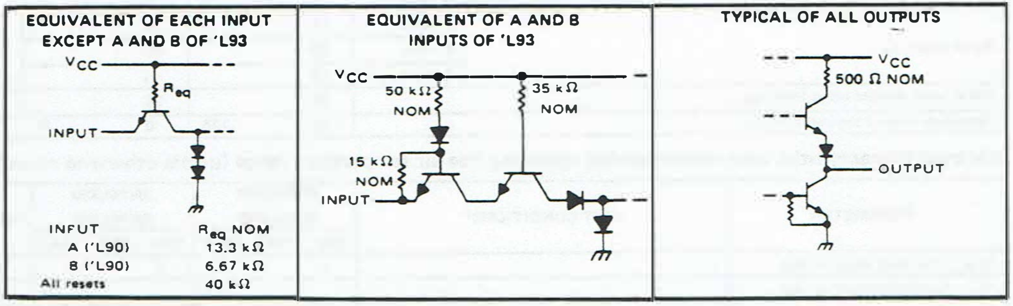
'90A, '92A, '93A


EQUIVALENT OF EACH INPUT

TYPICAL OF ALL OUTPUTS

INPUT	Req NOM
A	2.5 kΩ
B ('90A, '92A)	1.25 kΩ
B ('93A)	2.5 kΩ
All resets	6 kΩ

'L90, 'L93

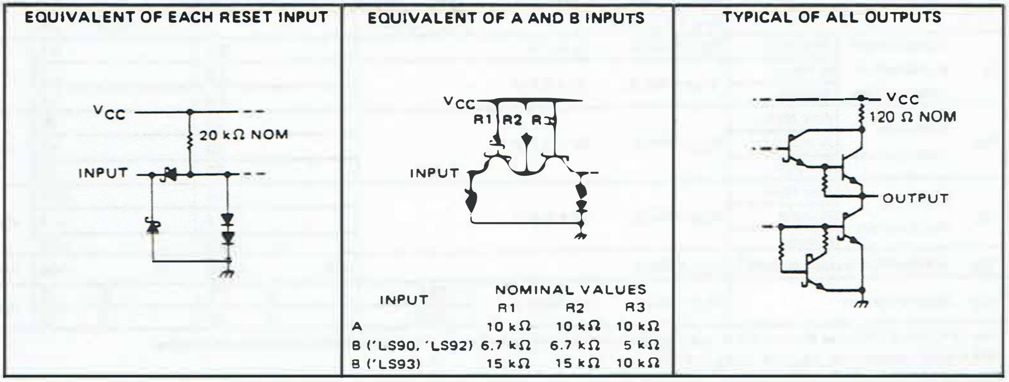

EQUIVALENT OF EACH INPUT EXCEPT A AND B OF 'L93

EQUIVALENT OF A AND B INPUTS OF 'L93

TYPICAL OF ALL OUTPUTS

INPUT	Req NOM
A ('L90)	13.3 kΩ
B ('L90)	6.67 kΩ
All resets	40 kΩ

'LS90, 'LS92, 'LS93


EQUIVALENT OF EACH RESET INPUT

EQUIVALENT OF A AND B INPUTS

TYPICAL OF ALL OUTPUTS

INPUT	NOMINAL VALUES R1	R2	R3
A	10 kΩ	10 kΩ	10 kΩ
B ('LS90, 'LS92)	6.7 kΩ	6.7 kΩ	5 kΩ
B ('LS93)	15 kΩ	15 kΩ	10 kΩ

TYPES SN54LS90, SN54LS92, SN54LS93, SN74LS90, SN74LS92, SN74LS93 DECADE, DIVIDE-BY-TWELVE, AND BINARY COUNTERS

REVISED OCTOBER 1976

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 4)		7 V
Input voltage:	R inputs	7 V
	A and B inputs	5.5 V
Operating free-air temperature range:	SN54LS' Circuits	−55°C to 125°C
	SN74LS' Circuits	0°C to 70°C
Storage temperature range		−65°C to 150°C

NOTE 4: Voltage values are with respect to network ground terminal.

recommended operating conditions

		SN54LS90 SN54LS92 SN54LS93			SN74LS90 SN74LS92 SN74LS93			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}		4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}				−400			−400	µA
Low-level output current, I_{OL}				4			8	mA
Count frequency, f_{count} (see Figure 1)	A input	0		32	0		32	MHz
	B input	0		16	0		16	
Pulse width, t_w	A input	15			15			ns
	B input	30			30			
	Reset inputs	15			15			
Reset inactive-state setup time, t_{su}		25			25			ns
Operating free-air temperature, T_A		−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS[†]		SN54LS90 SN54LS92			SN74LS90 SN74LS92			UNIT
					MIN	TYP[‡]	MAX	MIN	TYP[‡]	MAX	
V_{IH}	High-level input voltage				2			2			V
V_{IL}	Low-level input voltage						0.7			0.8	V
V_{IK}	Input clamp voltage		V_{CC} = MIN, I_I = −18 mA				−1.5			−1.5	V
V_{OH}	High-level output voltage		V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL} max, I_{OH} = −400 µA		2.5	3.4		2.7	3.4		V
V_{OL}	Low-level output voltage		V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL} max,	I_{OL} = 4 mA[¶]		0.25	0.4		0.25	0.4	V
				I_{OL} = 8 mA[¶]					0.35	0.5	
I_I	Input current at maximum input voltage	Any reset	V_{CC} = MAX, V_I = 7 V				0.1			0.1	mA
		A input	V_{CC} = MAX, V_I = 5.5 V				0.2			0.2	
		B input					0.4			0.4	
I_{IH}	High-level input current	Any reset	V_{CC} = MAX, V_I = 2.7 V				20			20	µA
		A input					40			40	
		B input					80			80	
I_{IL}	Low-level input current	Any reset	V_{CC} = MAX, V_I = 0.4 V				−0.4			−0.4	mA
		A input					−2.4			−2.4	
		B input					−3.2			−3.2	
I_{OS}	Short-circuit output current[§]		V_{CC} = MAX		−20		−100	−20		−100	mA
I_{CC}	Supply current		V_{CC} = MAX, See Note 3	'LS90		9	15		9	15	mA
				'LS92		9	15		9	15	

[†] For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.

[‡] All typical values are at V_{CC} = 5 V, T_A = 25°C.

[§] Not more than one output should be shorted at a time, and duration of the short-circuit should not exceed one second.

[¶] Q_A outputs are tested at specified I_{OL} plus the limit value of I_{IL} for the B input. This permits driving the B input while maintaining full fan-out capability.

NOTE 3: I_{CC} is measured with all outputs open, both R_0 inputs grounded following momentary connection to 4.5 V, and all other inputs grounded.

TYPES SN54LS90, SN54LS92, SN54LS93, SN74LS90, SN74LS92, SN74LS93
DECADE, DIVIDE-BY-TWELVE, AND BINARY COUNTERS

REVISED OCTOBER 1976

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS[†]		SN54LS93			SN74LS93			UNIT
					MIN	TYP[‡]	MAX	MIN	TYP[‡]	MAX	
V_{IH}	High-level input voltage				2			2			V
V_{IL}	Low-level input voltage						0.7			0.8	V
V_{IK}	Input clamp voltage		V_{CC} = MIN, $I_I = -18$ mA				−1.5			−1.5	V
V_{OH}	High-level output voltage		V_{CC} = MIN, V_{IH} = 2 V, $V_{IL} = V_{IL}$ max, $I_{OH} = -400$ μA		2.5	3.4		2.7	3.4		V
V_{OL}	Low-level output voltage		V_{CC} = MIN, V_{IH} = 2 V, $V_{IL} = V_{IL}$ max	I_{OL} = 4 mA[¶]		0.25	0.4		0.25	0.4	V
				I_{OL} = 8 mA[¶]					0.35	0.5	
I_I	Input current at maximum input voltage	Any reset	V_{CC} = MAX, V_I = 7 V				0.1			0.1	mA
		A or B input	V_{CC} = MAX, V_I = 5.5 V				0.2			0.2	
I_{IH}	High-level input current	Any reset	V_{CC} = MAX, V_I = 2.7 V				20			20	μA
		A or B input					40			80	
I_{IL}	Low-level input current	Any reset	V_{CC} = MAX, V_I = 0.4 V				−0.4			−0.4	mA
		A input					−2.4			−2.4	
		B input					−1.6			−1.6	
I_{OS}	Short-circuit output current[§]		V_{CC} = MAX		−20		−100	−20		−100	mA
I_{CC}	Supply current		V_{CC} = MAX, See Note 3			9	15		9	15	mA

[†] For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.

[‡] All typical values are at V_{CC} = 5 V, T_A = 25°C.

[§] Not more than one output should be shorted at a time, and duration of the short-circuit should not exceed one second.

[¶] Q_A outputs are tested at specified I_{OL} plus the limit value for I_{IL} for the B input. This permits driving the B input while maintaining full fan-out capability.

NOTE 3: I_{CC} is measured with all outputs open, both R_0 inputs grounded following momentary connection to 4.5 V, and all other inputs grounded.

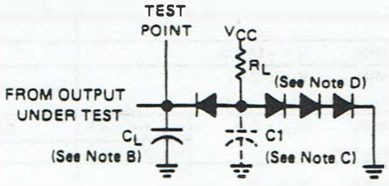
switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER[¶]	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	'LS90			'LS92			'LS93			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
f_{max}	A	Q_A	C_L = 15 pF, R_L = 2 kΩ See Figure 1	32	42		32	42		32	42		MHz
	B	Q_B		16			16			16			
t_{PLH}	A	Q_A			10	16		10	16		10	16	ns
t_{PHL}					12	18		12	18		12	18	
t_{PLH}	A	Q_D			32	48		32	48		46	70	ns
t_{PHL}					34	50		34	50		46	70	
t_{PLH}	B	Q_B			10	16		10	16		10	16	ns
t_{PHL}					14	21		14	21		14	21	
t_{PLH}	B	Q_C			21	32		10	16		21	32	ns
t_{PHL}					23	35		14	21		23	35	
t_{PLH}	B	Q_D			21	32		21	32		34	51	ns
t_{PHL}					23	35		23	35		34	51	
t_{PHL}	Set-to-0	Any			26	40		26	40		26	40	ns
t_{PLH}	Set-to-9	Q_A, Q_D			20	30							ns
t_{PHL}		Q_B, Q_C			26	40							

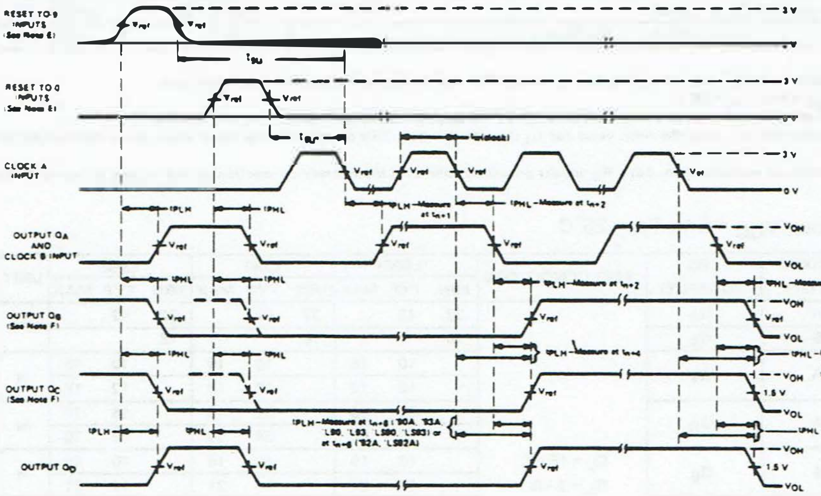
[¶] f_{max} ≡ maximum count frequency
t_{PLH} ≡ propagation delay time, low-to-high-level output
t_{PHL} ≡ propagation delay time, high-to-low-level output

TYPES SN5490A, SN5492A, SN5493A, SN54L90, SN54L93, SN54LS90, SN54LS92, SN54LS93, SN7490A, SN7492A, SN7493A, SN74L90, SN74L93, SN74LS90, SN74LS92, SN74LS93 DECADE, DIVIDE-BY-TWELVE, AND BINARY COUNTERS

PARAMETER MEASUREMENT INFORMATION



LOAD CIRCUIT



VOLTAGE WAVEFORMS

NOTES:
- A. Input pulses are supplied by a generator having the following characteristics:
  for '90A, '92A, '93A, $t_r \leq 5$ ns, $t_f \leq 5$ ns, PRR = 1 MHz, duty cycle = 50%, $Z_{out} \approx 50$ ohms;
  for 'L90, 'L93, $t_r \leq 15$ ns, $t_f \leq 15$ ns, PRR = 500 kHz, duty cycle = 50%, $Z_{out} \approx 50$ ohms;
  for 'LS90, 'LS92, 'LS93, $t_r \leq 15$ ns, $t_f \leq 5$ ns, PRR = 1 MHz, duty cycle = 50%, $Z_{out} \approx 50$ ohms.
- B. C_L includes probe and jig capacitance.
- C. C1 (30 pF) is applicable for testing 'L90 and 'L93.
- D. All diodes are 1N916 or 1N3064.
- E. Each reset input is tested separately with the other reset at 4.5 V.
- F. Reference waveforms are shown with dashed lines.
- G. For '90A, '92A, and '93A; $V_{ref} = 1.5$ V. For 'L90, 'L93, 'LS90, 'LS92, and 'LS93; $V_{ref} = 1.3$ V.

FIGURE 1

TTL
MSI

TYPES SN54LS138, SN54LS139, SN54S138, SN54S139, SN74LS138, SN74LS139, SN74S138, SN74S139
DECODERS/DEMULTIPLEXERS

BULLETIN NO. DL-S 7611804, DECEMBER 1972—REVISED OCTOBER 1976

- Designed Specifically for High-Speed:
  - Memory Decoders
  - Data Transmission Systems
- 'S138 and 'LS138 3-to-8-Line Decoders Incorporate 3 Enable Inputs to Simplify Cascading and/or Data Reception
- 'S139 and 'LS139 Contain Two Fully Independent 2-to-4-Line Decoders/Demultiplexers
- Schottky Clamped for High Performance

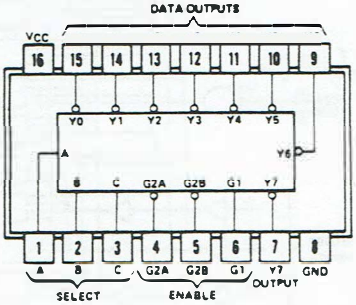
TYPE	TYPICAL PROPAGATION DELAY (3 LEVELS OF LOGIC)	TYPICAL POWER DISSIPATION
'LS138	22 ns	32 mW
'S138	8 ns	245 mW
'LS139	22 ns	34 mW
'S139	7.5 ns	300 mW

description

These Schottky-clamped TTL MSI circuits are designed to be used in high-performance memory-decoding or data-routing applications requiring very short propagation delay times. In high-performance memory systems these decoders can be used to minimize the effects of system decoding. When employed with high-speed memories utilizing a fast-enable circuit the delay times of these decoders and the enable time of the memory are usually less than the typical access time of the memory. This means that the effective system delay introduced by the Schottky-clamped system decoder is negligible.

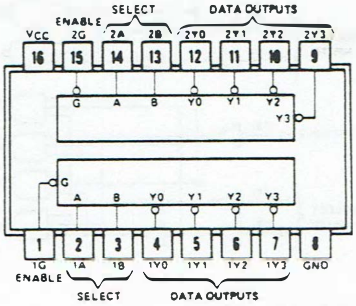
The 'LS138 and 'S138 decode one-of-eight lines dependent on the conditions at the three binary select inputs and the three enable inputs. Two active-low and one active-high enable inputs reduce the need for external gates or inverters when expanding. A 24-line decoder can be implemented without external inverters and a 32-line decoder requires only one inverter. An enable input can be used as a data input for demultiplexing applications.

SN54LS138, SN54S138 . . . J OR W PACKAGE
SN74LS138, SN74S138 . . . J OR N PACKAGE
(TOP VIEW)



positive logic: see function table

SN54LS139, SN54S139 . . . J OR W PACKAGE
SN74LS139, SN74S139 . . . J OR N PACKAGE
(TOP VIEW)



positive logic: see function table

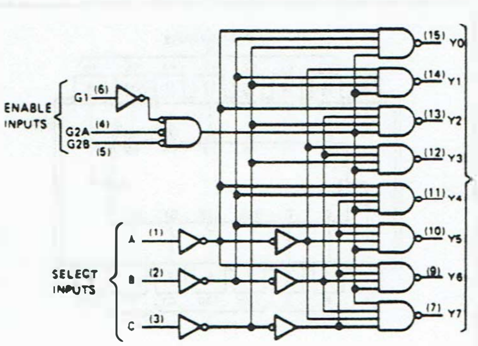
The 'LS139 and 'S139 comprise two individual two-line-to-four-line decoders in a single package. The active-low enable input can be used as a data line in demultiplexing applications.

All of these decoders/demultiplexers feature fully buffered inputs each of which represents only one normalized Series 54LS/74LS load ('LS138, 'LS139) or one normalized Series 54S/74S load ('S138, 'S139) to its driving circuit. All inputs are clamped with high-performance Schottky diodes to suppress line-ringing and simplify system design. Series 54LS and 54S devices are characterized for operation over the full military temperature range of −55°C to 125°C; Series 74LS and 74S devices are characterized for 0°C to 70°C industrial systems.

TYPES SN54LS138, SN54S138, SN54LS139, SN54S139 SN74LS138, SN74S138, SN74LS139, SN74S139 DECODERS/DEMULTIPLEXERS

functional block diagrams and logic

'LS138, 'S138

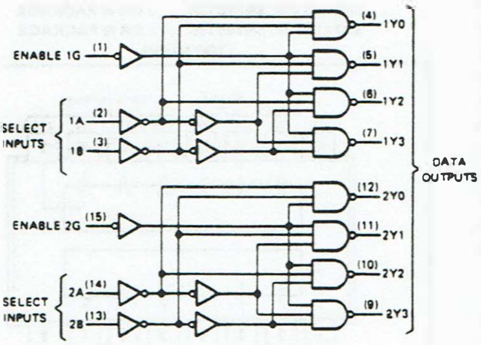


'LS138, 'S138
FUNCTION TABLE

INPUTS					OUTPUTS							
ENABLE		SELECT										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

*G2 = G2A + G2B
H = high level, L = low level, X = irrelevant

'LS139, 'S139



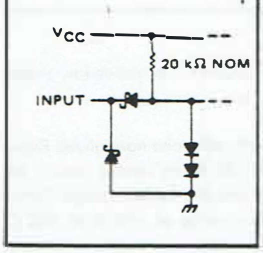
'LS139, 'S139
(EACH DECODER/DEMULTIPLEXER)
FUNCTION TABLE

INPUTS			OUTPUTS			
ENABLE	SELECT					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

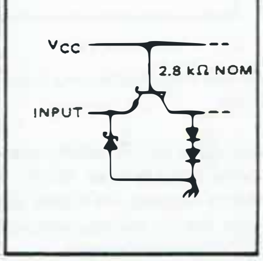
H = high level, L = low level, X = irrelevant

schematics of inputs and outputs

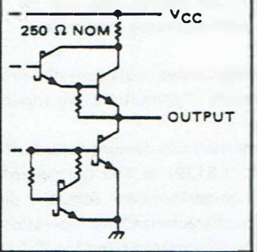
EQUIVALENT OF EACH INPUT OF 'LS138, 'LS139



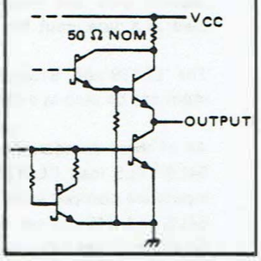
EQUIVALENT OF EACH INPUT OF 'S138, 'S139



TYPICAL OF OUTPUTS OF 'LS138, 'LS139



TYPICAL OF OUTPUTS OF 'S138, 'S139



TYPES SN54LS138, SN54LS139, SN74LS138, SN74LS139, DECODERS/DEMULTIPLEXERS

REVISED DECEMBER 1980

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1) . . . . 7 V
Input voltage . . . . 7 V
Operating free-air temperature range: SN54LS138, SN54LS139 Circuits . . . . . . −55°C to 125°C
SN74LS138, SN74LS139 Circuits . . . . . . 0°C to 70°C
Storage temperature range . . . . −65°C to 150°C

NOTE 1: Voltage values are with respect to network ground terminal.

recommended operating conditions

	SN54LS138 SN54LS139			SN74LS138 SN74LS139			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS†		SN54LS138 SN54LS139			SN74LS138 SN74LS139			UNIT
			MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH} High-level input voltage			2			2			V
V_{IL} Low-level input voltage					0.7			0.8	V
V_{IK} Input clamp voltage	V_{CC} = MIN, $I_I = -18$ mA				−1.5			−1.5	V
V_{OH} High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, $V_{IL} = V_{IL\,max}$, $I_{OH} = -400$ μA		2.5	3.4		2.7	3.4		V
V_{OL} Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, $V_{IL} = V_{IL\,max}$	I_{OL} = 4 mA		0.25	0.4		0.25	0.4	V
		I_{OL} = 8 mA					0.35	0.5	
I_I Input current at maximum input voltage	V_{CC} = MAX, V_I = 7 V				0.1			0.1	mA
I_{IH} High-level input current	V_{CC} = MAX, V_I = 2.7 V				20			20	μA
I_{IL} Low-level input current	V_{CC} = MAX, V_I = 0.4 V				−0.4			−0.4	mA
I_{OS} Short circuit output current §	V_{CC} = MAX	'LS138	−20		−100	−20		−100	mA
		'LS139	−6		−40	−5		−42	
I_{OS} Supply current	V_{CC} = MAX, Outputs enabled and open	'LS138		6.3	10		6.3	10	mA
		'LS139		6.8	11		6.8	11	

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.

‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.

§ Not more than one output should be shorted at a time.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER¶	FROM (INPUT)	TO (OUTPUT)	LEVELS OF DELAY	TEST CONDITIONS	SN54LS138 SN74LS138			SN54LS139 SN74LS139			UNIT
					MIN	TYP	MAX	MIN	TYP	MAX	
t_{PLH}	Binary Select	Any	2	C_L = 15 pF, R_L = 2 kΩ, See Note 2		13	20		13	20	ns
t_{PHL}						27	41		22	33	ns
t_{PLH}			3			18	27		18	29	ns
t_{PHL}						26	39		25	38	ns
t_{PLH}	Enable	Any	2			12	18		16	24	ns
t_{PHL}						21	32		21	32	ns
t_{PLH}			3			17	28				ns
t_{PHL}						25	38				ns

¶ t_{PLH} ≡ propagation delay time, low-to-high-level output; t_{PHL} ≡ propagation delay time, high-to-low-level output.

NOTE 2: Load circuits and waveforms are shown on page 3-11.

TYPES SN54S138, SN54S139, SN74S138, SN74S139 DECODERS/DEMULTIPLEXERS

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1)		7 V
Input voltage		5.5 V
Operating free-air temperature range:	SN54S138, SN54S139 Circuits	−55°C to 125°C
	SN74S138, SN74S139 Circuits	0°C to 70°C
Storage temperature range		−65°C to 150°C

NOTE 1: Voltage values are with respect to network ground terminal.

recommended operating conditions

	SN54S138 SN54S139			SN74S138 SN74S139			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−1			−1	mA
Low-level output current, I_{OL}			20			20	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS[†]		SN54S138 SN74S138			SN54S139 SN74S139			UNIT
				MIN	TYP[‡]	MAX	MIN	TYP[‡]	MAX	
V_{IH}	High-level input voltage			2			2			V
V_{IL}	Low-level input voltage					0.8			0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN, I_I = −18 mA				−1.2			−1.2	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OH} = −1 mA	SN54S'	2.5	3.4		2.5	3.4		V
			SN74S'	2.7	3.4		2.7	3.4		
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OL} = 20 mA				0.5			0.6	V
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 5.5 V				1			1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_I = 2.7 V				50			50	µA
I_{IL}	Low-level input current	V_{CC} = MAX, V_I = 0.5 V				−2			−2	mA
I_{OS}	Short-circuit output current[§]	V_{CC} = MAX		−40		−100	−40		−100	mA
I_{CC}	Supply current	V_{CC} = MAX, Outputs enabled and open			49	74		60	90	mA

[†] For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.

[‡] All typical values are at V_{CC} = 5 V, T_A = 25°C.

[§] Not more than one output should be shorted at a time, and duration of the short-circuit test should not exceed one second.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER[¶]	FROM (INPUT)	TO (OUTPUT)	LEVELS OF DELAY	TEST CONDITIONS	SN54S138, SN74S138			SN54S139 SN74S139			UNIT
					MIN	TYP	MAX	MIN	TYP	MAX	
t_{PLH}	Binary select	Any	2	C_L = 15 pF, R_L = 280 Ω, See Note 3		4.5	7		5	7.5	ns
t_{PHL}						7	10.5		6.5	10	
t_{PLH}			3			7.5	12		7	12	ns
t_{PHL}						8	12		8	12	
t_{PLH}	Enable	Any	2			5	8		6	8	ns
t_{PHL}						7	11		6.5	10	
t_{PLH}			3			7	11				ns
t_{PHL}						7	11				

[¶] t_{PLH} ≡ propagation delay time, low-to-high-level output
t_{PHL} ≡ propagation delay time, high-to-low-level output

NOTE 3: Load circuits and waveforms are shown on page 3-10.

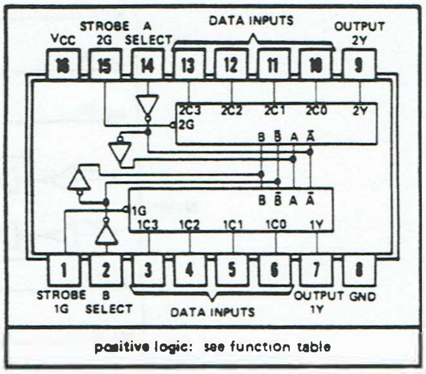
TTL
MSI

TYPES SN54153, SN54L153, SN54LS153, SN54S153,
SN74153, SN74L153, SN74LS153, SN74S153
DUAL 4-LINE-TO-1-LINE DATA SELECTORS/MULTIPLEXERS

BULLETIN NO. DL-S 7611852, DECEMBER 1972 — REVISED OCTOBER 1976

- Permits Multiplexing from N lines to 1 line
- Performs Parallel-to-Serial Conversion
- Strobe (Enable) Line Provided for Cascading (N lines to n lines)
- High-Fan-Out, Low-Impedance, Totem-Pole Outputs
- Fully Compatible with most TTL and DTL Circuits

SN54153, SN54LS153, SN54S153 . . . J OR W PACKAGE
SN54L153 . . . J PACKAGE
SN74153, SN74L153, SN74LS153, SN74S153 . . . J OR N PACKAGE
(TOP VIEW)



positive logic: see function table

TYPE	TYPICAL AVERAGE PROPAGATION DELAY TIMES FROM DATA	FROM STROBE	FROM SELECT	TYPICAL POWER DISSIPATION
'153	14 ns	17 ns	22 ns	180 mW
'L153	27 ns	34 ns	44 ns	90 mW
'LS153	14 ns	19 ns	22 ns	31 mW
'S153	6 ns	9.5 ns	12 ns	225 mW

FUNCTION TABLE

SELECT INPUTS		DATA INPUTS				STROBE	OUTPUT
B	A	C0	C1	C2	C3	G	Y
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

Select inputs A and B are common to both sections.
H = high level, L = low level, X = irrelevant

description

Each of these monolithic, data selectors/multiplexers contains inverters and drivers to supply fully complementary, on-chip, binary decoding data selection to the AND-OR-invert gates. Separate strobe inputs are provided for each of the two four-line sections.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

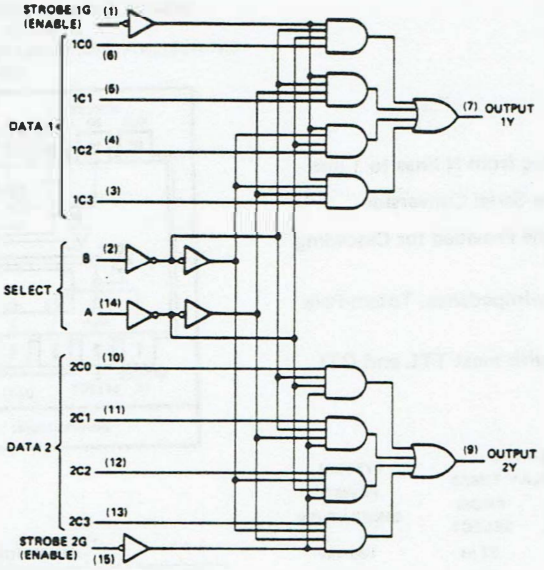
Supply voltage, V_{CC} (see Note 1) . . . . 7 V
Input voltage: '153, 'L153, 'S153 . . . . 5.5 V
'LS153 . . . . 7 V
Operating free-air temperature range: SN54', SN54L', SN54LS', SN54S' Circuits . . . . −55°C to 125°C
SN74', SN74L', SN74LS', SN74S' Circuits . . . . 0°C to 70°C
Storage temperature range . . . . −65°C to 150°C

NOTE 1: Voltage values are with respect to network ground terminal.

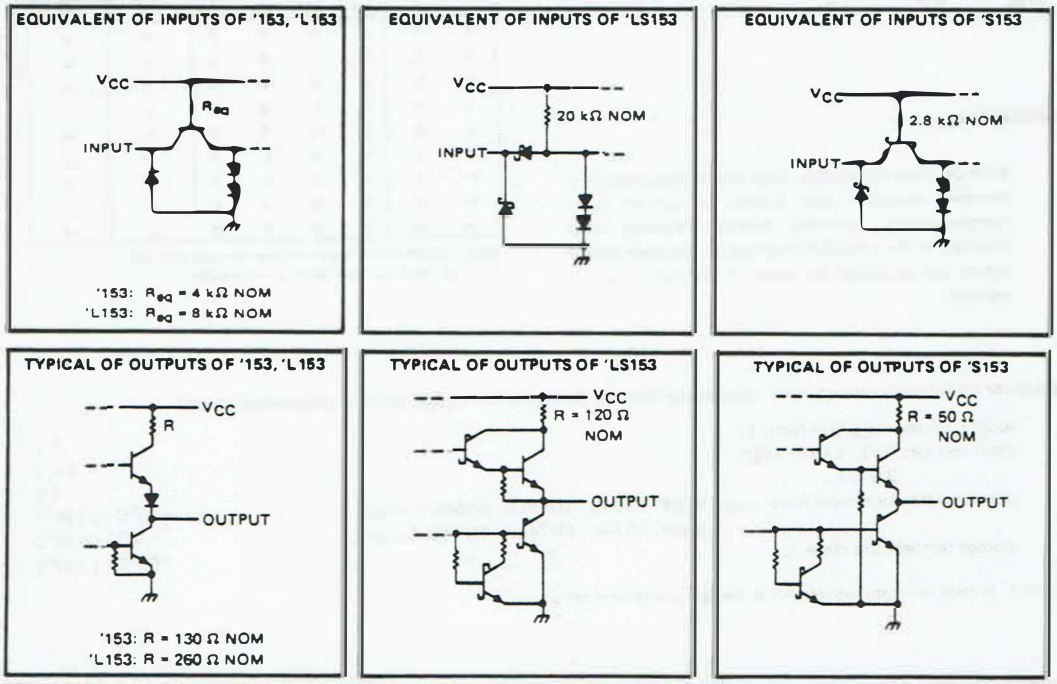
TYPES SN54153, SN54L153, SN54LS153, SN54S153, SN74153, SN74L153, SN74LS153, SN74S153 DUAL 4-LINE-TO-1-LINE DATA SELECTORS/MULTIPLEXERS

REVISED DECEMBER 1980

functional block diagram



schematics of inputs and outputs


EQUIVALENT OF INPUTS OF '153, 'L153
VCC
Req
INPUT
'153: Req = 4 kΩ NOM
'L153: Req = 8 kΩ NOM
EQUIVALENT OF INPUTS OF 'LS153
VCC
20 kΩ NOM
INPUT
EQUIVALENT OF INPUTS OF 'S153
VCC
2.8 kΩ NOM
INPUT
TYPICAL OF OUTPUTS OF '153, 'L153
VCC
R
OUTPUT
'153: R = 130 Ω NOM
'L153: R = 260 Ω NOM
TYPICAL OF OUTPUTS OF 'LS153
VCC
R = 120 Ω NOM
OUTPUT
TYPICAL OF OUTPUTS OF 'S153
VCC
R = 50 Ω NOM
OUTPUT

TYPES SN54LS153, SN74LS153
DUAL 4-LINE-TO-1-LINE DATA SELECTORS/MULTIPLEXERS

REVISED OCTOBER 1976

recommended operating conditions

	SN54LS153 MIN	SN54LS153 NOM	SN54LS153 MAX	SN74LS153 MIN	SN74LS153 NOM	SN74LS153 MAX	UNIT
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS†		SN54LS153 MIN	SN54LS153 TYP‡	SN54LS153 MAX	SN74LS153 MIN	SN74LS153 TYP‡	SN74LS153 MAX	UNIT
V_{IH} High-level input voltage			2			2			V
V_{IL} Low-level input voltage					0.7			0.8	V
V_{IK} Input clamp voltage	V_{CC} = MIN, I_I = −18 mA				−1.5			−1.5	V
V_{OH} High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL} max, I_{OH} = −400 μA		2.5	3.4		2.7	3.4		V
V_{OL} Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL} max	I_{OL} = 4 mA		0.25	0.4		0.25	0.4	V
		I_{OL} = 8 mA					0.35	0.6	
I_I Input current at maximum input voltage	V_{CC} = MAX, V_I = 7 V				0.1			0.1	mA
I_{IH} High-level input current	V_{CC} = MAX, V_I = 2.7 V				20			20	μA
I_{IL} Low-level input current	V_{CC} = MAX, V_I = 0.4 V				−0.4			−0.4	mA
I_{OS} Short-circuit output current§	V_{CC} = MAX		−20		−100	−20		−100	mA
I_{CCL} Supply current, output low	V_{CC} = MAX, See Note 2			6.2	10		6.2	10	mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.

‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.

§ Not more than one output should be shorted at a time and duration of short-circuit should not exceed one second.

NOTE 2: I_{CCL} is measured with the outputs open and all inputs grounded.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER¶	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Data	Y	C_L = 15 pF, R_L = 2 kΩ, See Note 4		10	15	ns
t_{PHL}	Data	Y			17	26	ns
t_{PLH}	Select	Y			19	29	ns
t_{PHL}	Select	Y			25	38	ns
t_{PLH}	Strobe	Y			16	24	ns
t_{PHL}	Strobe	Y			21	32	ns

¶ t_{PLH} ≡ propagation delay time, low-to-high-level output
t_{PHL} ≡ propagation delay time, high-to-low-level output

NOTE 4: Load circuits and voltage waveforms are shown on page 3-11.

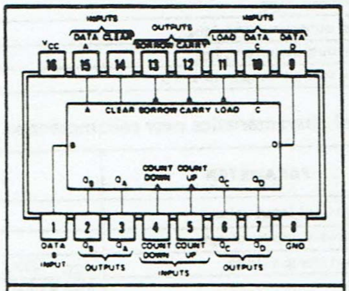
TYPES SN54192, SN54193, SN54L192, SN54L193, SN54LS192, SN54LS193 SN74192, SN74193, SN74L192, SN74L193, SN74LS192, SN74LS193 SYNCHRONOUS 4-BIT UP/DOWN COUNTERS (DUAL CLOCK WITH CLEAR)

BULLETIN NO. DL-S 7711828, DECEMBER 1972—REVISED AUGUST 1977

- Cascading Circuitry Provided Internally
- Synchronous Operation
- Individual Preset to Each Flip-Flop
- Fully Independent Clear Input

TYPES	TYPICAL MAXIMUM COUNT FREQUENCY	TYPICAL POWER DISSIPATION
'192, '193	32 MHz	325 mW
'L192, 'L193	7 MHz	43 mW
'LS192, 'LS193	32 MHz	95 mW

SN54', SN54LS' . . . J OR W PACKAGE
SN54L' . . . J PACKAGE
SN74', SN74L', SN74LS' . . . J OR N PACKAGE
(TOP VIEW)



logic: Low input to load sets Q_A = A, Q_B = B, Q_C = C, and Q_D = D

description

These monolithic circuits are synchronous reversible (up/down) counters having a complexity of 55 equivalent gates. The '192, 'L192, and 'LS192 circuits are BCD counters and the '193, 'L193 and 'LS193 are 4-bit binary counters. Synchronous operation is provided by having all flip-flops clocked simultaneously so that the outputs change coincidently with each other when so instructed by the steering logic. This mode of operation eliminates the output counting spikes which are normally associated with asynchronous (ripple-clock) counters.

The outputs of the four master-slave flip-flops are triggered by a low-to-high-level transition of either count (clock) input. The direction of counting is determined by which count input is pulsed while the other count input is high.

All four counters are fully programmable; that is, each output may be preset to either level by entering the desired data at the data inputs while the load input is low. The output will change to agree with the data inputs independently of the count pulses. This feature allows the counters to be used as modulo-N dividers by simply modifying the count length with the preset inputs.

A clear input has been provided which forces all outputs to the low level when a high level is applied. The clear function is independent of the count and load inputs. The clear, count, and load inputs are buffered to lower the drive requirements. This reduces the number of clock drivers, etc., required for long words.

These counters were designed to be cascaded without the need for external circuitry. Both borrow and carry outputs are available to cascade both the up- and down-counting functions. The borrow output produces a pulse equal in width to the count-down input when the counter underflows. Similarly, the carry output produces a pulse equal in width to the count-up input when an overflow condition exists. The counters can then be easily cascaded by feeding the borrow and carry outputs to the count-down and count-up inputs respectively of the succeeding counter.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

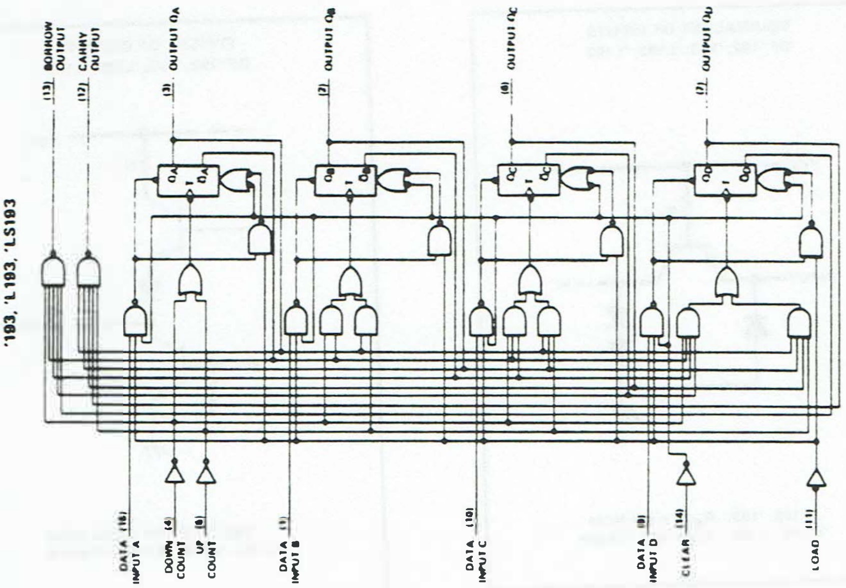
	SN54'	SN54L'	SN54LS'	SN74'	SN74L'	SN74LS'	UNIT
Supply voltage, V_{CC} (see Note 1)	7	8	7	7	8	7	V
Input voltage	5.5	5.5	7	5.5	5.5	7	V
Operating free-air temperature range	−55 to 125			0 to 70			°C
Storage temperature range	−65 to 150			−65 to 150			°C

NOTE 1: Voltage values are with respect to network ground terminal.

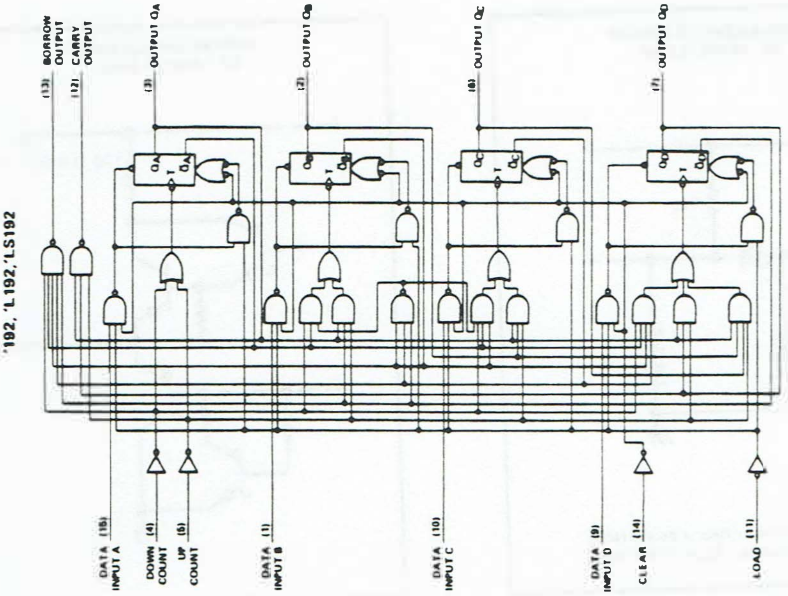
TYPES SN54192, SN54193, SN54L192, SN54L193, SN54LS192, SN54LS193, SN74192, SN74193, SN74L192, SN74L193, SN74LS192, SN74LS193 SYNCHRONOUS 4-BIT UP/DOWN COUNTERS (DUAL CLOCK WITH CLEAR)

functional block diagrams

'193, 'L193, 'LS193



'192, 'L192, 'LS192



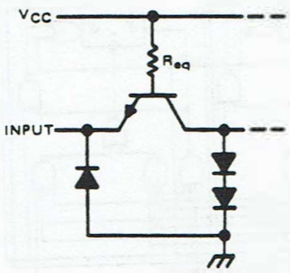
-▷- . . . Dynamic input activated by a transition from a high level to a low level.

TYPES SN54192, SN54193, SN54L192, SN54L193, SN54LS192, SN54LS193, SN74192, SN74193, SN74L192, SN74L193, SN74LS192, SN74LS193 SYNCHRONOUS 4-BIT UP/DOWN COUNTERS (DUAL CLOCK WITH CLEAR)

REVISED OCTOBER 1976

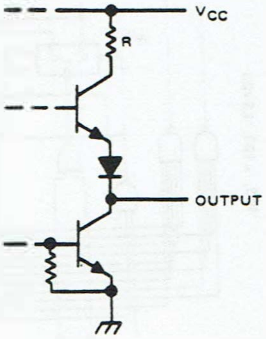
schematics of inputs and outputs

EQUIVALENT OF INPUTS
OF '192, '193, 'L192, 'L193



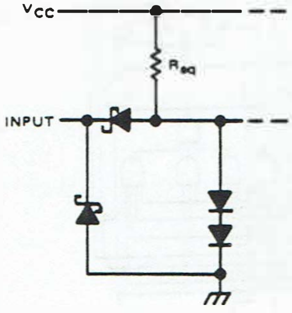
'192, '193: R_{eq} = 4 kΩ NOM
'L192, 'L193: R_{eq} = 40 kΩ NOM

TYPICAL OF OUTPUTS
OF '192, '193, 'L192, 'L193



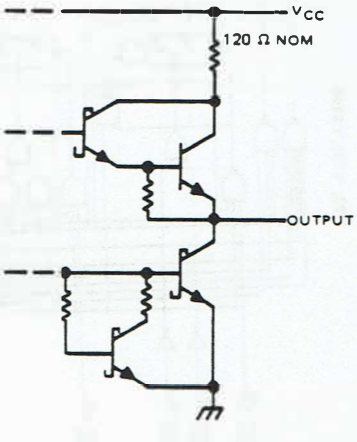
'192, '193: R = 130 Ω NOM
'L192, 'L193: R = 500 Ω NOM

EQUIVALENT OF INPUTS
OF 'LS192, 'LS193



Load Input: Req = 25 kΩ NOM
All other inputs: R_{eq} = 17 kΩ NOM

TYPICAL OF OUTPUTS
OF 'LS192, 'LS193



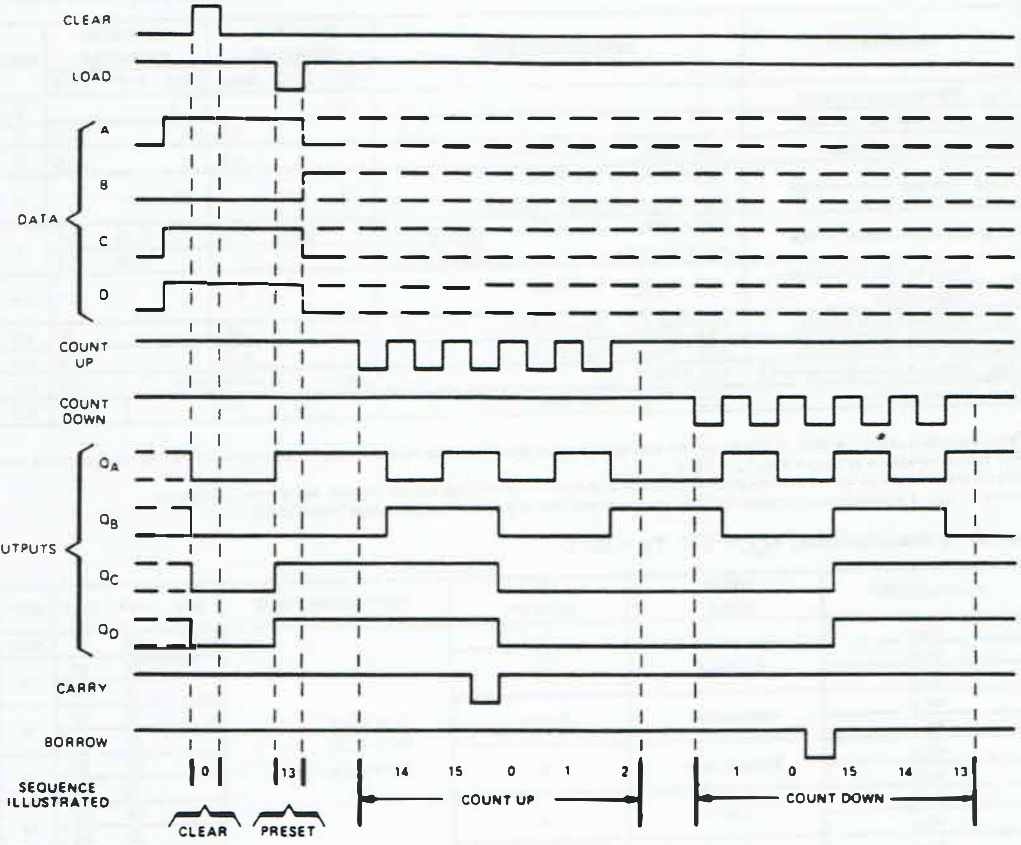
TYPES SN54193, SN54L193, SN54LS193, SN74193, SN74L193, SN74LS193 SYNCHRONOUS 4-BIT UP/DOWN COUNTERS (DUAL CLOCK WITH CLEAR)

'193, 'L193, 'LS193 BINARY COUNTERS

typical clear, load, and count sequences

Illustrated below is the following sequence:

1. Clear outputs to zero.
2. Load (preset) to binary thirteen.
3. Count up to fourteen, fifteen, carry, zero, one, and two.
4. Count down to one, zero, borrow, fifteen, fourteen, and thirteen.



NOTES: A. Clear overrides load, data, and count inputs.
B. When counting up, count-down input must be high; when counting down, count-up input must be high.

TYPES SN54LS192, SN54LS193, SN74LS192, SN74LS193
SYNCHRONOUS 4-BIT UP/DOWN COUNTERS (DUAL CLOCK WITH CLEAR)

REVISED DECEMBER 1980

recommended operating conditions

	SN54LS192 SN54LS193			SN74LS192 SN74LS193			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Clock frequency, f_{clock}	0		25	0		25	MHz
Width of any input pulse, t_w	20			20			ns
Clear inactive-state setup time	40			40			ns
Data setup time, t_{su} (see Figure 1)	20			20			ns
Data hold time, t_h	5			5			ns
Operating free-air temperature range, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS†	SN54LS192 SN54LS193			SN74LS192 SN74LS193			UNIT
		MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH} High-level input voltage		2			2			V
V_{IL} Low-level input voltage				0.7			0.8	V
V_{IK} Input clamp voltage	V_{CC} = MIN, I_I = −18 mA			−1.5			−1.5	V
V_{OH} High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL} max, I_{OH} = −400 μA	2.5	3.4		2.7	3.4		V
V_{OL} Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL} max, I_{OL} = 4 mA		0.25	0.4		0.15	0.4	V
	I_{OL} = 8 mA					0.35	0.5	
I_I Input current at maximum input voltage	V_{CC} = MAX, V_I = 7 V			0.1			0.1	mA
I_{IH} High-level input current	V_{CC} = MAX, V_I = 2.7 V			20			20	μA
I_{IL} Low-level input current	V_{CC} = MAX, V_I = 0.4 V			−0.4			−0.4	mA
I_{OS} Short-circuit output current§	V_{CC} = MAX	−20		−100	−20		−100	mA
I_{CC} Supply current	V_{CC} = MAX, See Note 2		19	34		19	34	mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable type.

‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.

§ Not more than one output should be shorted at a time ,and duration of the short-circuit should not exceed one second.

NOTE 2: I_{CC} is measured with all outputs open, clear and load inputs grounded, and all other inputs at 4.5 V.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER¶	FROM INPUT	TO OUTPUT	TEST CONDITIONS	MIN	TYP	MAX	UNIT
f_{max}			C_L = 15 pF, R_L = 2 kΩ, See Figures 1 and 2	25	32		MHz
t_{PLH}	Count-up	Carry			17	26	ns
t_{PHL}					18	24	
t_{PLH}	Count-down	Borrow			16	24	ns
t_{PHL}					15	24	
t_{PLH}	Either Count	Q			27	38	ns
t_{PHL}					30	47	
t_{PLH}	Load	Q			24	40	ns
t_{PHL}					25	40	
t_{PHL}	Clear	Q			23	35	ns

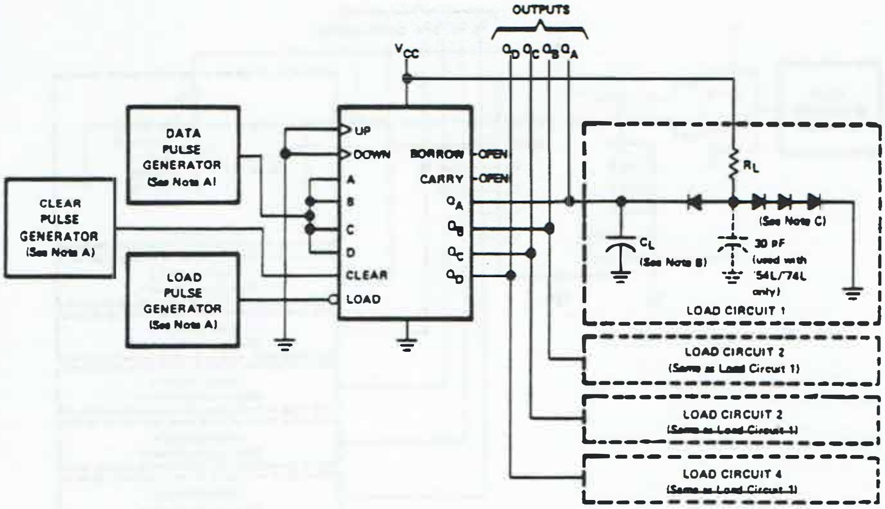
¶ f_{max} ≡ maximum clock frequency

t_{PLH} ≡ propagation delay time, low-to-high-level output

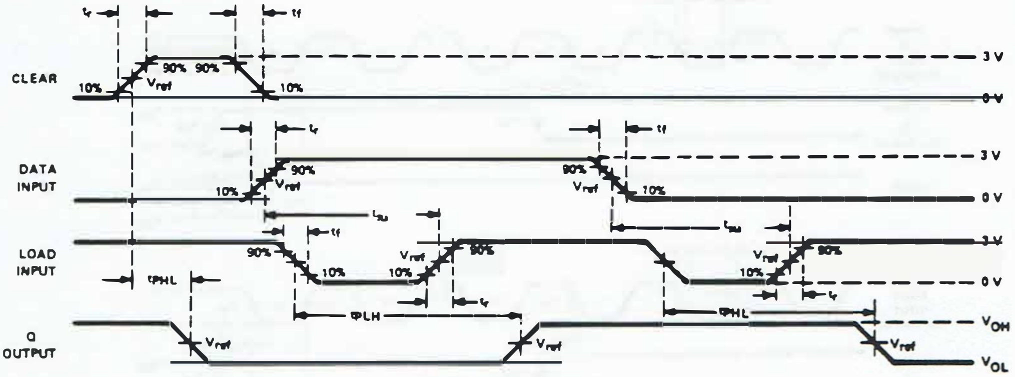
t_{PHL} ≡ propagation delay time, high-to-low-level output

TYPES SN54192, SN54193, SN54L192, SN54L193, SN54LS192, SN54LS193, SN74192, SN74193, SN74L192, SN74L193, SN74LS192, SN74LS193 SYNCHRONOUS 4-BIT UP/DOWN COUNTERS (DUAL CLOCK WITH CLEAR)

PARAMETER MEASUREMENT INFORMATION


OUTPUTS
VCC
QD QC QB QA
DATA PULSE GENERATOR (See Note A)
CLEAR PULSE GENERATOR (See Note A)
LOAD PULSE GENERATOR (See Note A)
UP
DOWN
A
B
C
D
CLEAR
LOAD
BORROW
CARRY
OPEN
OPEN
QA
QB
QC
QD
RL
CL (See Note B)
(See Note C)
30 pF (used with 54L/74L only)
LOAD CIRCUIT 1
LOAD CIRCUIT 2 (Same as Load Circuit 1)
LOAD CIRCUIT 2 (Same as Load Circuit 1)
LOAD CIRCUIT 4 (Same as Load Circuit 1)

TEST CIRCUIT



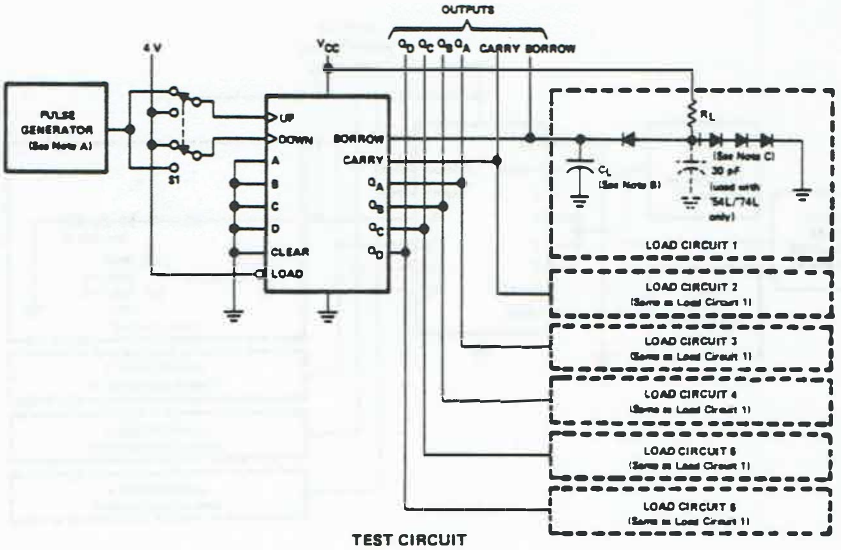
VOLTAGE WAVEFORMS

NOTES:
A. The pulse generators have the following characteristics: $Z_{out} \approx 50\ \Omega$ and for the data pulse generator PRR ≤ 500 kHz, duty cycle = 50%; for the load pulse generator PRR is two times data PRR, duty cycle = 50%.
B. C_L includes probe and jig capacitance.
C. Diodes are 1N3064 for '192, '193, 'LS192, and 'LS193; 1N916 for 'L192 and 'L193.
D. t_r and $t_f \leq 7$ ns for '192, '193, 'LS192, and 'LS193; ≤ 25 ns for 'L192 and 'L193.
E. V_{ref} is 1.5 volts for '192 and '193; 1.3 volts for 'L192, 'L193; 'LS192, and 'LS193.

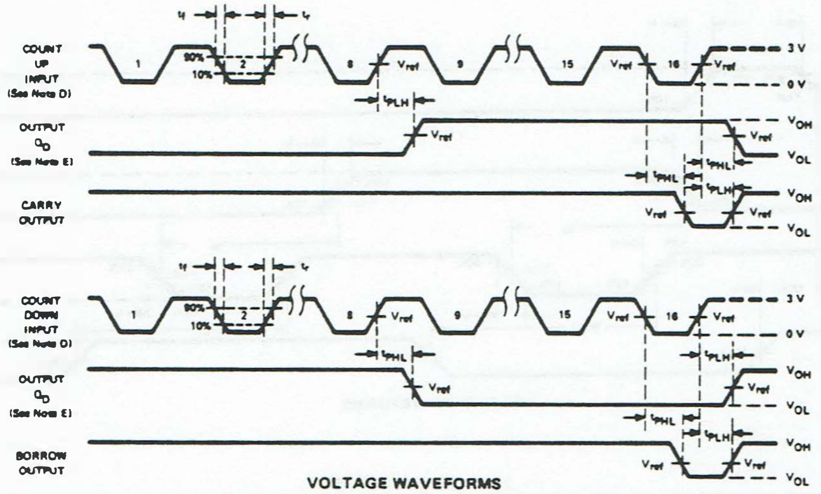
FIGURE 1—CLEAR, SETUP, AND LOAD TIMES

TYPES SN54192, SN54193, SN54L192, SN54L193, SN54LS192, SN54LS193, SN74192, SN74193, SN74L192, SN74L193, SN74LS192, SN74LS193 SYNCHRONOUS 4-BIT UP/DOWN COUNTERS (DUAL CLOCK WITH CLEAR)

PARAMETER MEASUREMENT INFORMATION


OUTPUTS
4 V
V_{CC}
Q_D Q_C Q_B Q_A CARRY BORROW
PULSE GENERATOR (See Note A)
S1
UP
DOWN
A
B
C
D
CLEAR
LOAD
BORROW
CARRY
Q_A
Q_B
Q_C
Q_D
R_L
C_L (See Note B)
(See Note C)
30 pF (used with 54L/74L only)
LOAD CIRCUIT 1
LOAD CIRCUIT 2 (Same as Load Circuit 1)
LOAD CIRCUIT 3 (Same as Load Circuit 1)
LOAD CIRCUIT 4 (Same as Load Circuit 1)
LOAD CIRCUIT 5 (Same as Load Circuit 1)
LOAD CIRCUIT 6 (Same as Load Circuit 1)

TEST CIRCUIT



VOLTAGE WAVEFORMS

NOTES:
- A. The pulse generator has the following characteristics: PRR ≤ 1 MHz, $Z_{out} \approx 50\ \Omega$, duty cycle = 50%.
- B. C_L includes probe and jig capacitance.
- C. Diodes are 1N3064 for '192, '193, 'LS192, and 'LS193; 1N916 for 'L192 and 'L193.
- D. Count-up and count-down pulse shown are for the '193, 'L193, and 'LS193 binary counters. Count cycle for '192, 'L192, and 'LS192 decade counters is 1 through 10.
- E. Waveforms for outputs Q_A, Q_B, and Q_C are omitted to simplify the drawing.
- F. t_r and $t_f \leq 7$ ns for '192, '193, 'LS192, and 'LS193; ≤ 25 ns for 'L192 and 'L193.
- G. V_{ref} is 1.5 volts for '192 and '193; 1.3 volts for 'L192, 'L193, LS192, and 'LS193.

FIGURE 2—PROPAGATION DELAY TIMES

Voltage to Frequency Frequency to Voltage Converters

9400 9401 9402

Features

Voltage-to-Frequency

- 1Hz to 100kHz Operation
- Choice of Guaranteed Linearity:

9401	.01%
9400	.05%
9402	.25%

- ±25ppm/°C Typ. Gain Temperature Stability
- Open Collector Output
- Output can Interface with any Form of Logic
- Pulse and Square Wave Outputs
- Programmable Scale Factor
- Low Power Dissipation (27mW Typ.)
- Single Supply Operation (8V to 15V)
- Dual Supply Operation (±4V to ±7.5V)
- Current or Voltage Input

Frequency-to-Voltage

- DC to 100kHz Operation
- Choice of Guaranteed Linearity:

9401	0.02%
9400	0.05%
9402	0.25%

- Op Amp Output
- Programmable Scale Factor
- High Input Impedance >10MΩ
- Accepts any Voltage Wave Shape

General Description

The Teledyne 9400 Series are low cost Voltage-to-Frequency converters combining Bipolar and CMOS technology on the same substrate. The converters accept a variable analog input signal and generate an output pulse train whose frequency is linearly proportional to the input voltage.

The 9400 Series can also be used as highly accurate Frequency-to-Voltage converters, accepting virtually any input frequency waveform and providing a linearly proportional voltage output.

A complete V/F or F/V system requires the addition of two capacitors and three resistors and reference voltage. In addition the 9400's operate on single or dual supply voltages.

Ordering Information

9400 CJ	14 Pin Plastic DIP	0°C to 70°C
9401 CJ	14 Pin Plastic DIP	0°C to 70°C
9402 CJ	14 Pin Plastic DIP	0°C to 70°C
9400 CN	14 Pin Ceramic DIP	-40°C to +85°C
9401 CN	14 Pin Ceramic DIP	-40°C to +85°C

Applications

Voltage-to-Frequency

- Temperature Sensing and Control
- µP Data Acquisition
- Instrumentation
- 13-Bit A/D Converters
- Digital Panel Meters
- Analog Data Transmission and Recording
- Phase Locked Loops
- Medical Isolation
- Transducer Encoding
- Alternate to 555 Astable Timer

Frequency-to-Voltage

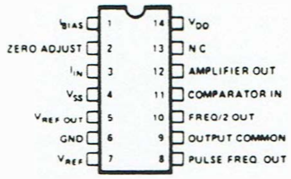
- Frequency Meters/Tachometer
- Speedometers
- Analog Data Transmission and Recording
- Medical Isolation
- Motor Control
- RPM Indicator
- FM Demodulation
- Frequency Multiplier/Divider
- Flow Measurement and Control

Absolute Maximum Ratings

Storage Temperature		-65°C to +150°C
Operating Temperature	J Package	0°C to 70°C
	N Package	-40°C to +85°C
$V_{DD} - V_{SS}$		18V
I_{IN}		10mA
V_{OUT} Max — V_{OUT} Common		25V
$V_{REF} - V_{SS}$		-1.5V
Package Dissipation		500mW
Lead Temperature (Soldering, 10 sec)		300°C

HANDLING PRECAUTIONS. The 9400 Series are CMOS Bipolar devices and must be handled correctly to prevent damage. Package and store only in conductive foam, anti-static tubes or other conductive material. Use proper anti-static handling procedures. Do not connect in circuits under "power on" conditions, as high transients may cause permanent damage.

Connection Diagram



NOTE: PIN 1 INDICATED BY ADJACENT DOT IN PACKAGE (INDENT OR END NOTCH (J PACKAGE)

9400, 9401, 9402

Electrical Characteristics, V/F Mode

Unless otherwise specified. V_{DD} = +5V. V_{SS} = —5V. V_{GND} = 0. V_{REF} = —5V. R_{BIAS} = 100KΩ. Full Scale = 10KHz. T_A = 25°C unless Full Temp Range is specified —40°C to +85°C for N package. 0°C to 70°C for J package

VOLTAGE-TO-FREQUENCY		9401			9400			9402				
Parameter	**Definition**	**Min.**	**Typ.**	**Max.**	**Min.**	**Typ.**	**Max.**	**Min.**	**Typ.**	**Max.**	**Units**	**Notes**
Accuracy Linearity 10KHz	Output Deviation from Straight Line between Normalized Zero and Full Scale Input		0 004	**0.01**		0 01	**0.05**		0 05	**0.25**	% Full Scale	
Linearity 100KHz			0 04	0 08		0 1	0 25		0 25	0 50	% Full Scale	
Gain Temperature Drift	Variation in Gain A due to Temperature Change		±25	±40		±25	±40		±50	±100	ppm/°C Full Scale	1
Gain Variance	Variation from Exact A Compensate by Trimming R_{IN}. V_{REF} or C_{REF}		±10			±10			±10		% of Nominal	
Zero Offset	Correction at Zero Adjust for Zero Output When Input is Zero		±10	±50		±10	±50		±20	±100	mV	2
Zero Temperature Drift	Variation in Zero Offset Due to Temperature Change		±25	±50		±25	±50		±50	±100	μV/°C	1
Analog Inputs I_{IN} Full Scale	Full Scale Analog Input Current to Achieve Specified Accuracy		10			10			10		μA	
I_{IN} Overrange	Overrange Current			50			50			50	μA	
Response Time	Settling Time to 0 01% Full Scale		2			2			2		Cycles	
Digital Outputs V_{SAT} @ I_{OL} 10μA	Logical 0 Output Voltage			0 4			0 4			0 4	V	3
V_{OUT} Max -- V_{OUT} Common	Voltage Range between Output and Common			18 0			18 0			18 0	V	4
Pulse Frequency Output Width			3 0			3 0			3 0		μsec	
Supply Current I_{DD} Quiescent N Package J Package	Current Required from Positive Supply During Operation		2 0 2 0	4 0 6 0		2 0 2 0	4 0 6 0		3 0	10 0	mA mA	9
I_{SS} Quiescent N Package J Package	Current Required from Negative Supply During Operation		-1 5 -1 5	-4 0 -6 0		-1 5 -1 5	-4 0 -6 0		-3 0	-10 0	mA mA	10
V_{DD} Supply	Operating Range of Positive Supply	4 0		7 5	4 0		7 5	4 0		7 5	V	
V_{SS} Supply	Operating Range of Negative Supply	-4 0		-7 5	-4 0		-7 5	-4 0		-7 5	V	
Reference Voltage V_{REF} — V_{SS}	Range of Voltage Reference Input	-1 0			-1 0			-1 0			V	

NOTES

1 Full temperature range
2 I_{IN} = 0
3 Full temperature range I_{OUT} 10mA
4 I_{OUT} = 10μA
5 10Hz to 100KHz
6 5μs min positive pulse width and 0 5μs min negative pulse width
7 T_r t_f 20ns
8 R_L 2KΩ
9 Full temperature range V_{IN} -0 1V
10 V_{IN} -0 1V
11 I_{IN} connects the summing junction of an operational amplifier Voltage sources cannot be attached directly but must be buffered by external resistors

9400, 9401, 9402

V/F Circuit Description

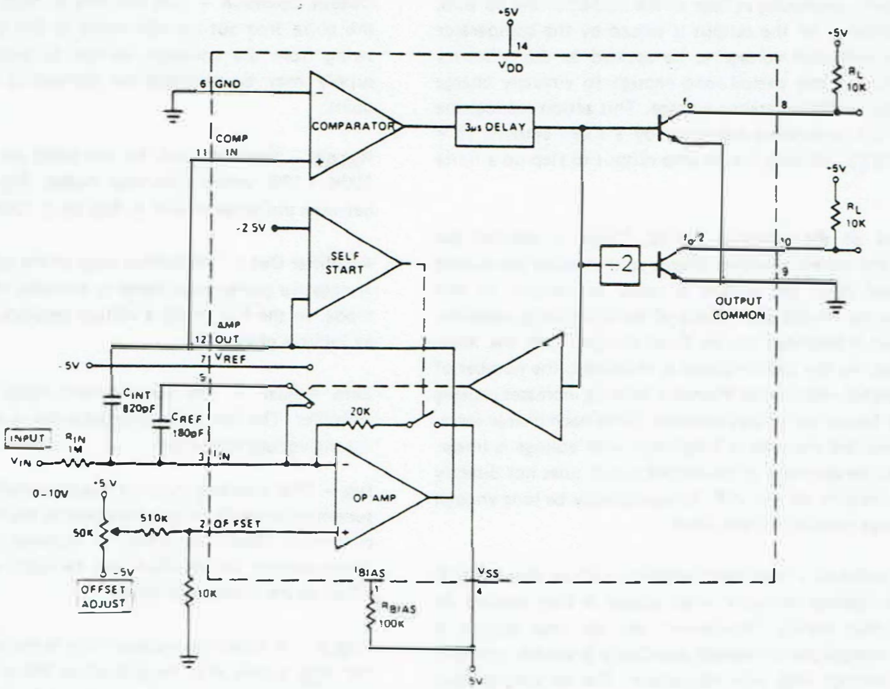
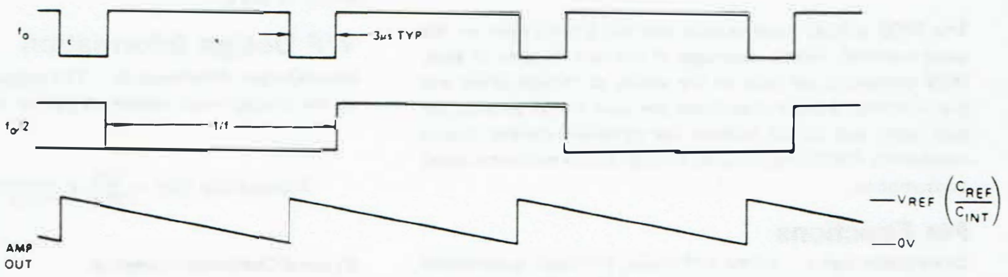


Figure 1. 10Hz to 10KHz V/F Converter



1. To adjust f_{min}, set $V_{IN} = 10mV$ and adjust the 50K offset for 10Hz out.
2. To adjust f_{max}, set $V_{IN} = 10V$ and adjust R_{IN} or V_{REF} for 10KHz out.
3. To increase $f_{OUT}MAX$ to 100KHz change C_{REF} to 27pF and C_{INT} to 75pF.
4. For high performance applications use high stability components for R_{IN}, C_{REF}, V_{REF} (metal film resistors and glass film capacitors). Also separate the output ground (Pin 9) from the input ground (Pin 6).

Figure 2. Output Waveforms

V/F Circuit Description (Contd.)

The Teledyne 9400 V/F Converter operates on the principal of charge balancing. The input voltage (V_{IN}) is converted to a current (I_{IN}) by the input resistor. This current is then converted to a charge by the integrating capacitor and shows up as a linearly decreasing voltage at the output of the op amp. The zero crossing of the output is sensed by the comparator causing the reference voltage to be applied to the reference capacitor for a time period long enough to virtually charge the capacitor to the reference voltage. This action reduces the charge on the integrating capacitor by a fixed amount ($q = C_{REF} \times V_{REF}$) causing the op amp output to step up a finite amount.

At the end of the charging period, C_{REF} is shorted out dissipating the stored reference charge so that when the output again crosses zero, the system is ready to recycle. In this manner, the continued discharging of the integrating capacitor by the input is balanced out by fixed charges from the reference voltage. As the input voltage is increased, the number of reference pulses required to maintain balance increases causing the output frequency to also increase. Since each charge increment is fixed the increase in frequency with voltage is linear. In addition, the accuracy of the output pulses does not directly effect the linearity of the V/F. It must simply be long enough for full charge transfer to take place.

The 9400 contains a "self-start" circuit to assure that the V/F will always operate properly when power is first applied. In the event that during "Power-on" the op amp output is below the comparator threshold and C_{REF} is already charged, a positive voltage step will not occur. The op amp output will continue to decrease until it crosses the −2.5 volt threshold of the "self-start" comparator. When this happens a resistor is connected to the op amp input causing the output to quickly go positive until the 9400 is once again in its normal operating mode.

The 9400 utilizes both bipolar and MOS transistors on the same substrate, taking advantage of the best features of each. MOS transistors are used at the inputs to reduce offset and bias currents. Bipolar transistors are used in the op amp, for high gain, and on all outputs for excellent current driving capabilities. CMOS logic is used throughout to minimize power consumption.

Pin Functions

Comparator Input – In the V/F mode, this input is connected to the amplifier output (pin 12) and triggers the 3μsec pulse delay when the input voltage passes its threshold. In the F/V mode, the input frequency is applied to the comparator input.

Pulse Freq Out – This output is an open-collector bipolar transistor providing a pulse waveform whose frequency is proportional to the input voltage. This output requires a pull up resistor and interfaces directly with MOS, CMOS and TTL logic.

Freq/2 Out – This output is an open-collector bipolar transistor providing a square wave that is one-half the frequency of the pulse frequency output. This output requires a pull up resistor and interfaces directly with MOS, CMOS, and TTL logic.

Output Common – The emitters of both the freq/2 out and the pulse freq out are connected to this pin. An output level swing from the collector voltage to ground or to the V_{SS} supply may be obtained by connecting to the appropriate point.

R_{BIAS} – Specifications for the 9400 are based on R_{BIAS} = 100K ±10% unless otherwise noted. R_{BIAS} may be varied between the range of $82K \leq R_{BIAS} \leq 120K$.

Amplifier Out – The output stage of the operational amplifier. A negative going ramp signal is available at this pin in the V/F mode. In the F/V mode a voltage proportional to the frequency input is generated.

Zero Adjust – The non-inverting input of the operational amplifier. The low frequency set point is detemined by adjusting the voltage at this pin.

I_{IN} – The inverting input of the operational amplifier and the summing junction when connected in the V/F mode. An input current of 10μA is specified for nominal full scale but an over range current up to 50μA can be used without detrimental effect to the circuit operation.

V_{REF} – A reference voltage from either a precision source or the V_{SS} supply may be applied to this pin. Accuracy will be dependent on the voltage regulation and temperature characteristics of the circuitry.

V_{REF} OUT – The charging current for C_{REF} is derived from the internal circuitry and switched by the break-before-make switch to this pin.

V/F Design Information

Input/Output Relationships – The output frequency is related to the analog input voltage (V_{IN}) by the transfer equation:

$$\text{Frequency Out} = \frac{V_{IN}}{R_{IN}} \times \frac{1}{V_{REF} \quad C_{REF}} = f_o$$

External Component Selection

R_{IN} – The value of this component is chosen to give a full scale input current of approximately 10μA.

$$R_{IN} \cong \frac{V_{IN}\ \text{FULL SCALE}}{10\mu A}$$

Example:

$$R_{IN} \cong \frac{10V}{10\mu A} = 1M\Omega$$

9400, 9401, 9402

V/F Design Information (Contd.)

Note that the value is an approximation, and the exact relationship is defined by the transfer equation. In practice, the value of R_{IN} typically would be trimmed to obtain full scale frequency at V_{IN} FULL SCALE (see adjustment procedure). Metal film resistors with 1% tolerance or better are recommended for high accuracy applications because of their thermal stability and low noise generation.

C_{INT} – Exact value not critical but is related to C_{REF} by the relationship:

$$3C_{REF} \leq C_{INT} \leq 10C_{REF}$$

Improved stability and linearity is obtained when $C_{INT} \geq 4C_{REF}$. Low leakage types are recommended although mica and ceramic devices can be used in applications where their temperature limits are not exceeded. Locate as close as possible to pins 12 and 3.

C_{REF} – Exact value not critical and may be used to trim the full scale frequency (see input/output relation). Glass film or air trimmer capacitors are recommended because of their stability and low leakage. Locate as close as possible to pins 5 and 3.

V_{DD}, V_{SS} – Power supplies of ±5V are recommended. For high accuracy requirements 0.05% line and load regulation and 0.1μF disc decoupling capacitors located near the pins are recommended.

Adjustment Procedure – Figure 1 shows a circuit for trimming the zero location. Full scale may be trimmed by adjusting R_{IN}, V_{REF}, or C_{REF}. Recommended procedure is as follows for a 10KHz full scale frequency.

1. Set V_{IN} to 10mV and trim the zero adjust circuit to obtain a 10Hz output frequency.
2. Set V_{IN} to 10.000V and trim either R_{IN}, V_{REF}, or C_{REF} to obtain a 10KHz output frequency.

If adjustments are performed in this order, there should be no interaction and they should not have to be repeated.

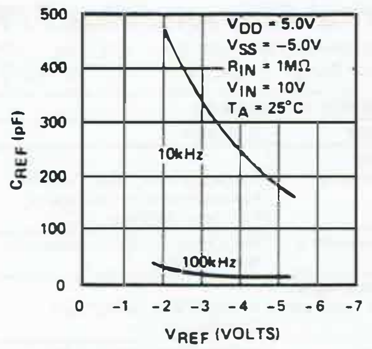
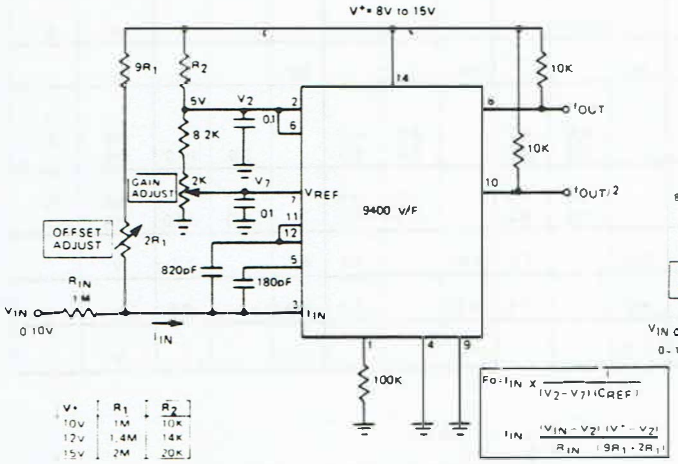


Figure 3. Recommended C_{REF} vs V_{REF}

V/F Single Supply Operation



$$F_O = I_{IN} \times \frac{1}{(V_2 - V_7)(C_{REF})}$$

$$I_{IN} = \frac{(V_{IN} - V_2)}{R_{IN}} - \frac{(V^+ - V_2)}{(9R_1 + 2R_1)}$$

Figure 4. Fixed Voltage—Single Supply Operation

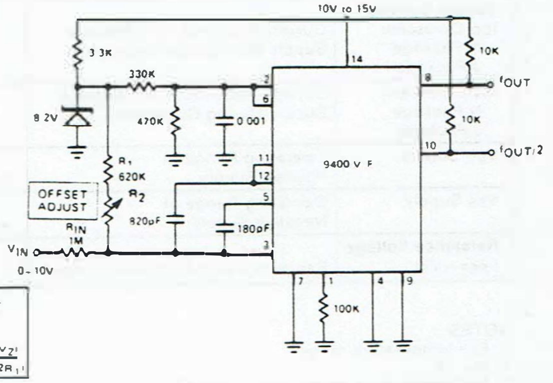


Figure 5. Variable Voltage—Single Supply Operation

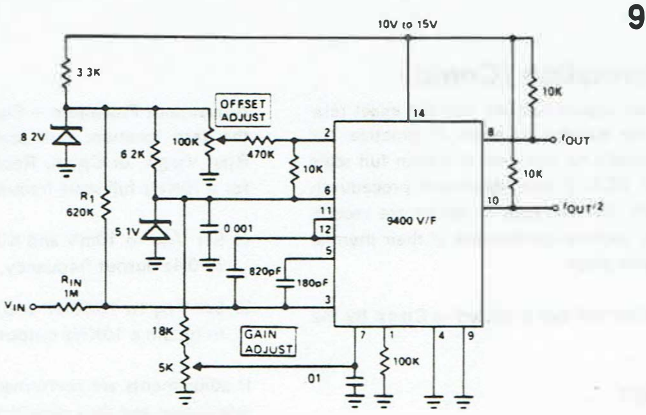


Figure 6. Single Variable Supply Voltage with Offset and Gain Adjust

Electrical Characteristics, F/V Mode

Unless otherwise specified V_{DD} +5V V_{SS} −5V V_{GND} 0 V_{REF} −5V R_{BIAS} 100KΩ Full Scale 10KHz T_A = 25°C unless Full Temp Range is specified −40°C to +85°C for N package 0°C to 70°C for J package

FREQUENCY-TO-VOLTAGE		9401			9400			9402				
Parameter	Definition	Min.	Typ.	Max.	Min.	Typ.	Max.	Min.	Typ.	Max.	Units	Notes
Accuracy Non-Linearity	Deviation from Ideal Transfer Function as a Percentage of Full Scale Voltage		0 01	**0.02**		0 02	**0.05**		0 05	**0.25**	% Full Scale	5
Input Frequency Range	Frequency Range for Specified Non-Linearity	10		100K	10		100K	10		100K	Hz	6
Frequency Inputs Positive Excursion	Voltage Required to Turn Comparator On	0 4		V_{DD}	0 4		V_{DD}	0 4		V_{DD}	V	7
Negative Excursion	Voltage Required to Turn Comparator Off	-0 4		-2V	-0 4		-2V	-0 4		-2V	V	7
Min Positive Pulse Width	Time between Threshold Crossings		5 0			5 0			5 0		μs	7
Min Negative Pulse Width	Time between Threshold Crossings		0 5			0 5			0 5		μs	7
Input Impedance		10			10			10			MΩ	
Analog Outputs Output Voltage	Voltage Range of Op Amp Output for Specified Non-Linearity		V_{DD}-1			V_{DD}-1			V_{DD}-1		V	8
Output Loading	Resistive Loading at Output of Op Amp	2K			2K			2K			Ω	
Supply Current I_{DD} Quiescent N Package J Package	Current Required from Positive Supply During Operation		2 0 2 0	4 0 6 0		2 0 2 0	4 0 6 0		 3 0	 10 0	mA mA	9
I_{SS} Quiescent N Package J Package	Current Required from Negative Supply During Operation		-1 5 -1 5	-4 0 -6 0		-1 5 -1 5	-4 0 -6 0		 -3 0	 -10 0	mA mA	10
V_{DD} Supply	Operating Range of Positive Supply	4 0		7 5	4 0		7 5	4 0		7 5	V	
V_{SS} Supply	Operating Range of Negative Supply	-4 0		-7 5	-4 0		-7 5	-4 0		-7 5	V	
Reference Voltage $V_{REF}-V_{SS}$	Range of Voltage Reference Input	-1 0			-1 0			-1 0			V	

NOTES
1 Full temperature range
2 I_{IN} 0
3 Full temperature range I_{OUT} 10mA
4 I_{OUT} 10μA
5 10Hz to 100KHz
6 5μs min positive pulse width and 0 5μs min negative pulse width
7 T_r t_f 20ns
8 R_L 2KΩ
9 Full temperature range V_{IN} -0 1V
10 V_{IN} -0 1V
11 I_{IN} connects the summing junction of an operational amplifier Voltage sources cannot be attached directly but must be buffered by external resistors

F/V Circuit Description

The 9400, when used as a frequency to voltage converter, generates an output voltage which is linearly proportional to the input frequency waveform.

Each zero crossing at the comparator's input causes a precise amount of charge ($q = C_{REF} \times V_{REF}$) to be dispensed into the op amp's summing junction. This charge in turn flows through the feedback resistor generating voltage pulses at the output of the op amp. A capacitor (C_{INT}) across R_{INT} averages these pulses into a DC voltage which is linearly proportional to the input frequency.

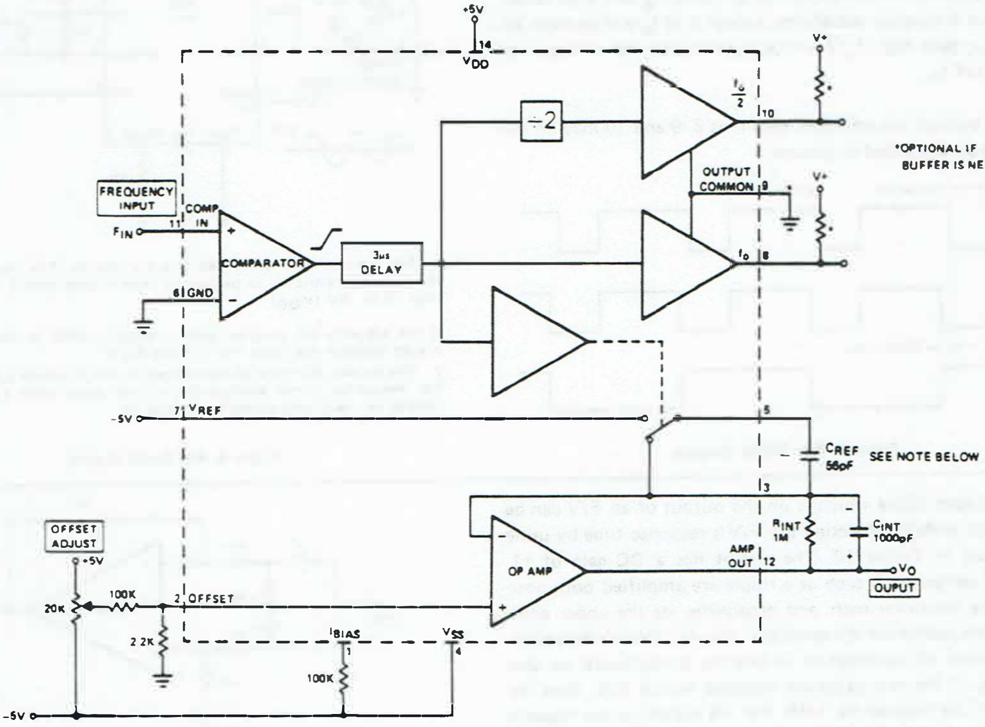


Figure 7. DC — 10KHz F/V Converter

F/V Design Information

Input/Output Relationships — The output voltage is related to the input frequency (F_{IN}) by the transfer equation:

$$V_{OUT} = \left[V_{REF}\, C_{REF}\, R_{INT}\right] F_{IN}$$

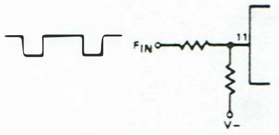
The response time to a change in F_{IN} is equal to $(R_{INT}\, C_{INT})$. The amount of ripple on V_{OUT} is inversely proportional to C_{INT} and the Input Frequency.

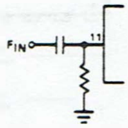
C_{INT} can be increased to lower the ripple. 1µF to 100µF are perfectly acceptable values for low frequencies.

When 9400 is used in the single supply mode, V_{REF} is defined as the voltage difference between Pin 7 and Pin 2.

Input Voltage Levels — The input signal must cross through zero in order to trip the comparator. In order to overcome the hysteresis the amplitude must be greater than ±200mV.

If only a unipolar input signal (F_{IN}) is available, it is recommended that either an offset circuit using resistor be used or that the signal be coupled in via a capacitor.





Note: C_{REF} should be increased for lower F_{IN} max. Adjust C_{REF} so that V_O is approximately 2.5 to 3.0 volts for the maximum input frequency. When F_{IN} max is less than 1 kHz, the duty cycle should be greater than 20% to insure that C_{REF} is fully charged and discharged.

F/V Design Information (Contd.)

For 100KHz maximum input R_{INT} should be decreased to 100KΩ.

Input Buffer – f_O and $f_O/2$ are not used in the F/V mode. However, these outputs may be useful for some applications, such as a buffer to feed additional circuitry. f_O will then follow the input frequency waveform; except that f_O will go high 3μs after F_{IN} goes high. $f_O/2$ will be square wave with a frequency of one half f_O.

If these outputs are not used, then Pins 8, 9 and 10 may be left floating or connected to ground.

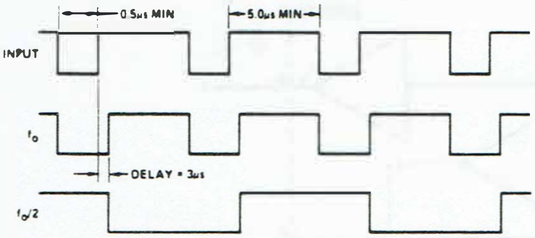
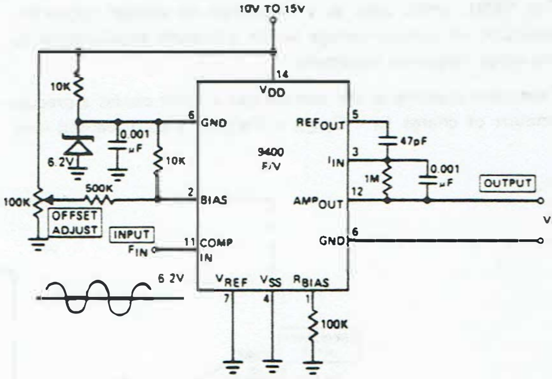


Figure 8. F/V Digital Outputs



1. The input is now referenced to 6.2 V (Pin 6). The input signal must therefore be restricted to be greater than 4 volts (Pin 6 –2V) and less than 10 to 15V (V_{DD}).

If the signal is AC coupled then a resistor (100K to 10MΩ) must be placed between the input (Pin 11) and Pin 6.

2. The output will now be referenced to Pin 6 which is at 6.2 V(V_Z). For frequency meter applications a 1mA meter with a series scaling resistor can be placed across Pins 6 and 12.

Figure 9. F/V Single Supply

The sawtooth ripple which is on the output of an F/V can be eliminated without affecting the F/V's response time by using the circuit in Figure 10. The circuit has a DC gain of +1. Any AC components such as a ripple are amplified both positively, via the lower path, and negatively, via the upper path. When both paths have the same gain, the AC ripple is cancelled. The amount of cancellation is directly proportional to gain matching. If the two paths are matched within 10%, then the ripple will be lowered by 1/10. For 1% matching, the ripple is lowered by 1/100. The 10K potentiometer is used to make the gain equal in both paths. This circuit is insensitive to both frequency changes and to signal wave shape.

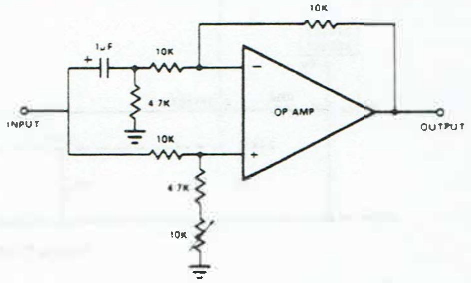
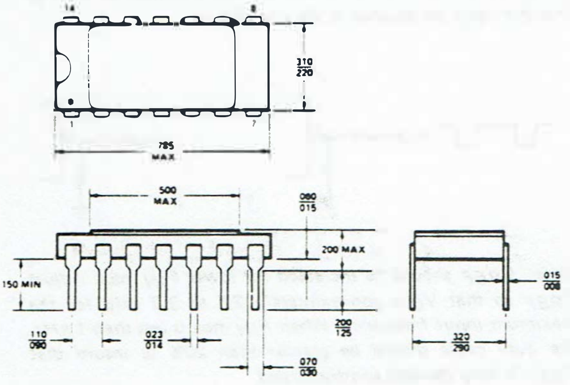
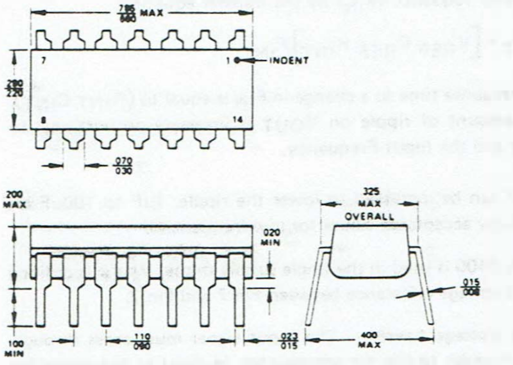


Figure 10. F/V Ripple Eliminator

Physical Dimensions



N Package
14-Pin Ceramic DIP



J Package
14-Pin Plastic DIP

National Semiconductor

August 1985

ADC0801, ADC0802, ADC0803, ADC0804, ADC0805 8-Bit μP Compatible A/D Converters

General Description

The ADC0801, ADC0802, ADC0803, ADC0804 and ADC0805 are CMOS 8-bit successive approximation A/D converters which use a differential potentiometric ladder—similar to the 256R products. These converters are designed to allow operation with the NSC800 and INS8080A derivative control bus, and TRI-STATE® output latches directly drive the data bus. These A/Ds appear like memory locations or I/O ports to the microprocessor and no interfacing logic needed.

A new differential analog voltage input allows increasing the common-mode rejection and offsetting the analog zero input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

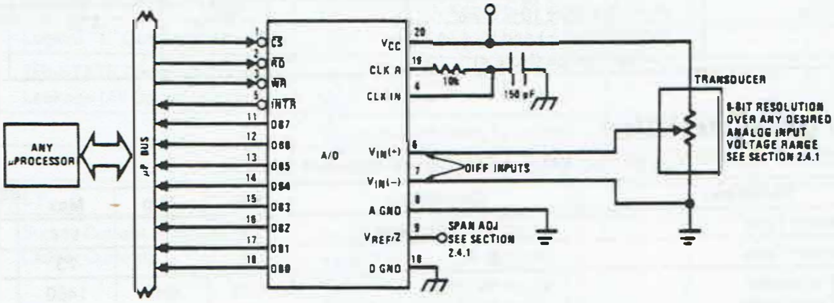
Features

- Compatible with 8080 μP derivatives—no interfacing logic needed - access time - 135 ns
- Easy interface to all microprocessors, or operates "stand alone"
- Differential analog voltage inputs
- Logic inputs and outputs meet both MOS and T²L voltage level specifications
- Works with 2.5V (LM336) voltage reference
- On-chip clock generator
- 0V to 5V analog input voltage range with single 5V supply
- No zero adjust required
- 0.3" standard width 20-pin DIP package
- Operates ratiometrically or with 5 V_{DC}, 2.5 V_{DC}, or analog span adjusted voltage reference

Key Specifications

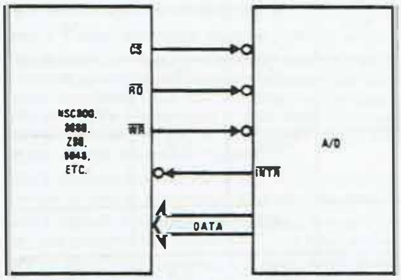
■ Resolution	8 bits
■ Total error	±¼ LSB, ±½ LSB and ±1 LSB
■ Conversion time	100 μs

Typical Applications



TL/H/5671-1

8080 Interface



TL/H/5671-31

Error Specification (Includes Full-Scale, Zero Error, and Non-Linearity)			
Part Number	**Full-Scale Adjusted**	**$V_{REF}/2 = 2.500$ V_{DC} (No Adjustments)**	**$V_{REF}/2$ = No Connection (No Adjustments)**
ADC0801	±¼ LSB		
ADC0802		±½ LSB	
ADC0803	±½ LSB		
ADC0804		±1 LSB	
ADC0805			±1 LSB

TRI-STATE® is a registered trademark of National Semiconductor Corp.
Z-80® is a registered trademark of Zilog Corp.

©1985 National Semiconductor Corporation TL/H/5671

Absolute Maximum Ratings (Notes 1 & 2)

Supply Voltage (V_{CC}) (Note 3)	6.5V
Voltage	
Logic Control Inputs	−0.3V to +18V
At Other Input and Outputs	−0.3V to (V_{CC}+0.3V)
Storage Temperature Range	−65°C to +150°C
Package Dissipation at T_A = 25°C	875 mW
Lead Temp. (Soldering, 10 seconds)	300°C

Operating Conditions (Notes 1 & 2)

Temperature Range	$T_{MIN} \le T_A \le T_{MAX}$
ADC0801/02LJ	$-55°C \le T_A \le +125°C$
ADC0801/02/03/04LCJ	$-40°C \le T_A \le +85°C$
ADC0801/02/03/05LCN	$0°C \le T_A \le +70°C$
ADC0804LCN	$0°C \le T_A \le +70°C$
Range of V_{CC}	4.5 V_{DC} to 6.3 V_{DC}

Electrical Characteristics

The following specifications apply for V_{CC} = 5 V_{DC}, $T_{MIN} \le T_A \le T_{MAX}$ and f_{CLK} = 640 kHz unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
ADC0801: Total Adjusted Error (Note 8)	With Full-Scale Adj. (See Section 2.5.2)			± 1/4	LSB
ADC0802: Total Unadjusted Error (Note 8)	$V_{REF}/2$ = 2.500 V_{DC}			± 1/2	LSB
ADC0803: Total Adjusted Error (Note 8)	With Full-Scale Adj. (See Section 2.5.2)			± 1/2	LSB
ADC0804: Total Unadjusted Error (Note 8)	$V_{REF}/2$ = 2.500 V_{DC}			± 1	LSB
ADC0805: Total Unadjusted Error (Note 8)	$V_{REF}/2$-No Connection			± 1	LSB
$V_{REF}/2$ Input Resistance (Pin 9)	ADC0801/02/03/05 ADC0804 (Note 9)	2.5 1.0	8.0 1.3		kΩ kΩ
Analog Input Voltage Range	(Note 4) V(+) or V(−)	Gnd-0.05		V_{CC}+0.05	V_{DC}
DC Common-Mode Error	Over Analog Input Voltage Range		± 1/16	± 1/8	LSB
Power Supply Sensitivity	V_{CC} = 5 V_{DC} ± 10% Over Allowed $V_{IN}(+)$ and $V_{IN}(-)$ Voltage Range (Note 4)		± 1/16	± 1/8	LSB

AC Electrical Characteristics

The following specifications apply for V_{CC} = 5 V_{DC} and T_A = 25°C unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
T_C	Conversion Time	f_{CLK} = 640 kHz (Note 6)	103		114	μs
T_C	Conversion Time	(Note 5, 6)	66		73	$1/f_{CLK}$
f_{CLK}	Clock Frequency Clock Duty Cycle	V_{CC} = 5V, (Note 5) (Note 5)	100 40	640	1460 60	kHz %
CR	Conversion Rate in Free-Running Mode	$\overline{\text{INTR}}$ tied to $\overline{\text{WR}}$ with $\overline{\text{CS}}$ = 0 V_{DC}, f_{CLK} = 640 kHz			8770	conv/s
$t_{W(\overline{WR})L}$	Width of $\overline{\text{WR}}$ Input (Start Pulse Width)	$\overline{\text{CS}}$ = 0 V_{DC} (Note 7)	100			ns
t_{ACC}	Access Time (Delay from Falling Edge of $\overline{\text{RD}}$ to Output Data Valid)	C_L = 100 pF		135	200	ns
t_{1H}, t_{0H}	TRI-STATE Control (Delay from Rising Edge of $\overline{\text{RD}}$ to Hi-Z State)	C_L = 10 pF, R_L = 10k (See TRI-STATE Test Circuits)		125	200	ns
t_{WI}, t_{RI}	Delay from Falling Edge of $\overline{\text{WR}}$ or $\overline{\text{RD}}$ to Reset of $\overline{\text{INTR}}$			300	450	ns
C_{IN}	Input Capacitance of Logic Control Inputs			5	7.5	pF
C_{OUT}	TRI-STATE Output Capacitance (Data Buffers)			5	7.5	pF

Electrical Characteristics (Continued)

The following specifications apply for $V_{CC} = 5V_{DC}$ and $T_{MIN} \le T_A \le T_{MAX}$, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
CONTROL INPUTS [Note: CLK IN (Pin 4) is the input of a Schmitt trigger circuit and is therefore specified separately]						
V_{IN} (1)	Logical "1" Input Voltage (Except Pin 4 CLK IN)	$V_{CC} = 5.25\ V_{DC}$	2.0		15	V_{DC}
V_{IN} (0)	Logical "0" Input Voltage (Except Pin 4 CLK IN)	$V_{CC} = 4.75\ V_{DC}$			0.8	V_{DC}
I_{IN} (1)	Logical "1" Input Current (All Inputs)	$V_{IN} = 5\ V_{DC}$		0.005	1	μA_{DC}
I_{IN} (0)	Logical "0" Input Current (All Inputs)	$V_{IN} = 0\ V_{DC}$	-1	-0.005		μA_{DC}
CLOCK IN AND CLOCK R						
V_T+	CLK IN (Pin 4) Positive Going Threshold Voltage		2.7	3.1	3.5	V_{DC}
V_T-	CLK IN (Pin 4) Negative Going Threshold Voltage		1.5	1.8	2.1	V_{DC}
V_H	CLK IN (Pin 4) Hysteresis $(V_T+) - (V_T-)$		0.6	1.3	2.0	V_{DC}
V_{OUT} (0)	Logical "0" CLK R Output Voltage	$I_O = 360\ \mu A$ $V_{CC} = 4.75\ V_{DC}$			0.4	V_{DC}
V_{OUT} (1)	Logical "1" CLK R Output Voltage	$I_O = -360\ \mu A$ $V_{CC} = 4.75\ V_{DC}$	2.4			V_{DC}
DATA OUTPUTS AND $\overline{\text{INTR}}$						
V_{OUT} (0)	Logical "0" Output Voltage Data Outputs $\overline{\text{INTR}}$ Output	 $I_{OUT} = 1.6$ mA, $V_{CC} = 4.75\ V_{DC}$ $I_{OUT} = 1.0$ mA, $V_{CC} = 4.75\ V_{DC}$			 0.4 0.4	 V_{DC} V_{DC}
V_{OUT} (1)	Logical "1" Output Voltage	$I_O = -360\ \mu A$, $V_{CC} = 4.75\ V_{DC}$	2.4			V_{DC}
V_{OUT} (1)	Logical "1" Output Voltage	$I_O = -10\ \mu A$, $V_{CC} = 4.75\ V_{DC}$	4.5			V_{DC}
I_{OUT}	TRI-STATE Disabled Output Leakage (All Data Buffers)	$V_{OUT} = 0\ V_{DC}$ $V_{OUT} = 5\ V_{DC}$	-3		 3	μA_{DC} μA_{DC}
I_{SOURCE}		V_{OUT} Short to Gnd, $T_A = 25°C$	4.5	6		mA_{DC}
I_{SINK}		V_{OUT} Short to V_{CC}, $T_A = 25°C$	9.0	16		mA_{DC}
POWER SUPPLY						
I_{CC}	Supply Current (Includes Ladder Current)	$f_{CLK} = 640$ kHz, $V_{REF}/2 = NC$, $T_A = 25°C$ and $\overline{CS} = "1"$ ADC0801/02/03/05 ADC0804 (Note 9)		 1.1 1.9	 1.8 2.5	 mA mA

Note 1: Absolute maximum ratings are those values beyond which the life of the device may be impaired.

Note 2: All voltages are measured with respect to Gnd, unless otherwise specified. The separate A Gnd point should always be wired to the D Gnd.

Note 3: A zener diode exists, internally, from V_{CC} to Gnd and has a typical breakdown voltage of 7 V_{DC}.

Note 4: For $V_{IN}(-) \ge V_{IN}(+)$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input (see block diagram) which will forward conduct for analog input voltages one diode drop below ground or one diode drop greater than the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct–especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

Note 5: Accuracy is guaranteed at f_{CLK} = 640 kHz. At higher clock frequencies accuracy can degrade. For lower clock frequencies, the duty cycle limits can be extended so long as the minimum clock high time interval or minimum clock low time interval is no less than 275 ns.

Note 6: With an asynchronous start pulse, up to 8 clock periods may be required before the internal clock phases are proper to start the conversion process. The start request is internally latched, see *Figure 2* and section 2.0.

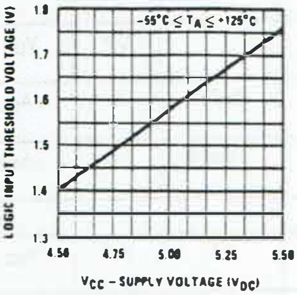
Note 7: The $\overline{CS}$ input is assumed to bracket the $\overline{WR}$ strobe input and therefore timing is dependent on the $\overline{WR}$ pulse width. An arbitrarily wide pulse width will hold the converter in a reset mode and the start of conversion is initiated by the low to high transition of the $\overline{WR}$ pulse (see timing diagrams).

Note 8: None of these A/Ds requires a zero adjust (see section 2.5.1). To obtain zero code at other analog input voltages see section 2.5 and *Figure 5*.

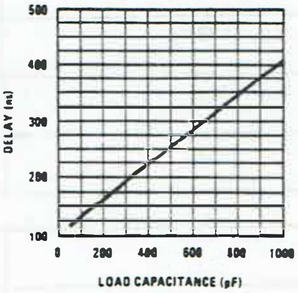
Note 9: For ADC0804LCJ typical value of $V_{REF}/2$ input resistance is 8 kΩ and of I_{CC} is 1.1 mA.

Typical Performance Characteristics

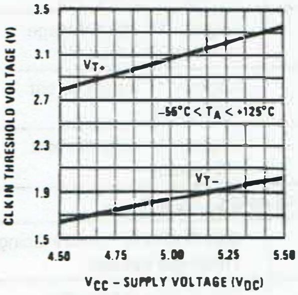
Logic Input Threshold Voltage vs. Supply Voltage



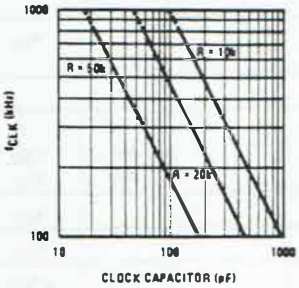
Delay From Falling Edge of RD to Output Data Valid vs. Load Capacitance



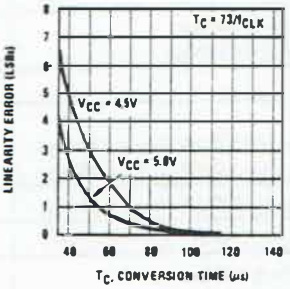
CLK IN Schmitt Trip Levels vs. Supply Voltage



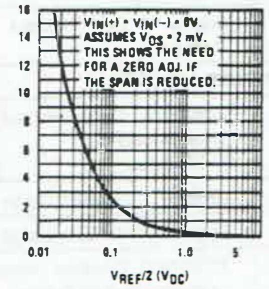
f_{CLK} vs. Clock Capacitor



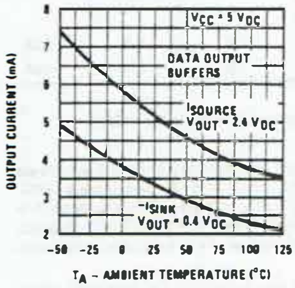
Full-Scale Error vs Conversion Time



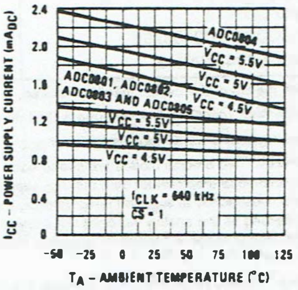
Effect of Unadjusted Offset Error vs. $V_{REF}/2$ Voltage



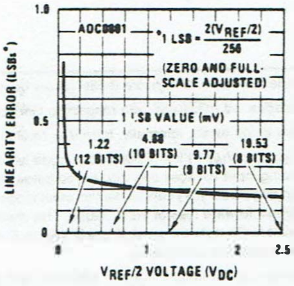
Output Current vs Temperature



Power Supply Current vs Temperature (Note 9)

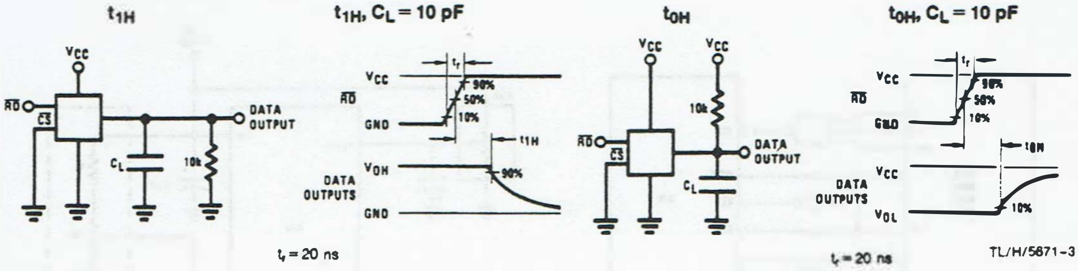


Linearity Error at Low $V_{REF}/2$ Voltages



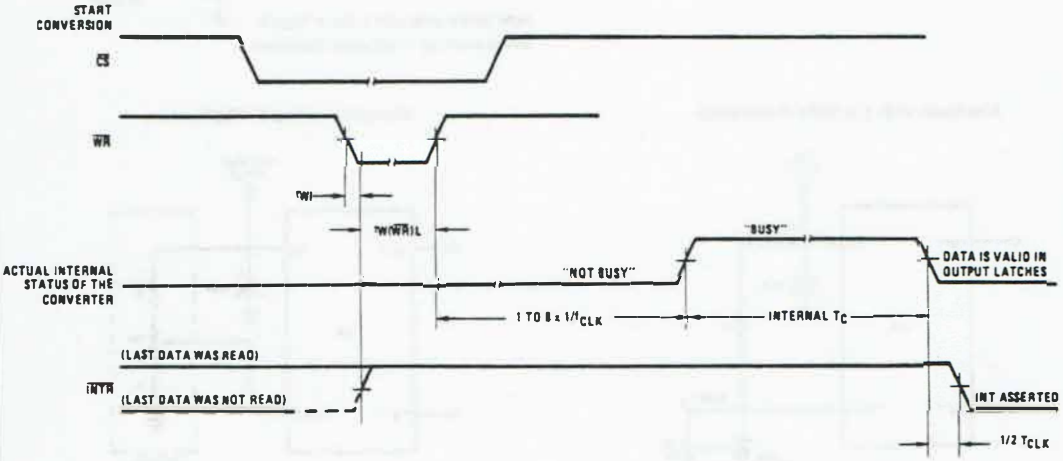
TL/H/5671-2

TRI-STATE Test Circuits and Waveforms

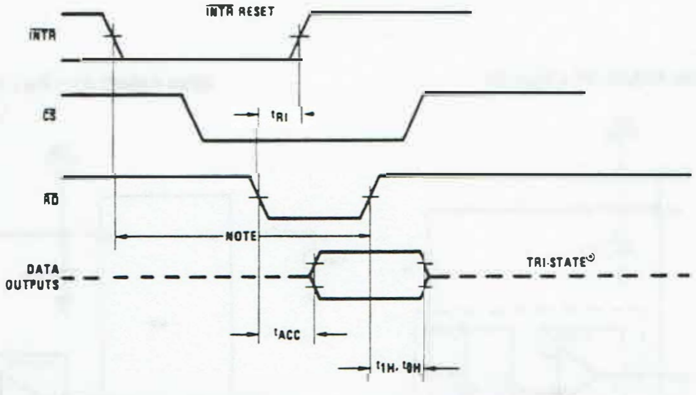


TL/H/5671-3

Timing Diagrams (All timing is measured from the 50% voltage points)



Output Enable and Reset INTR

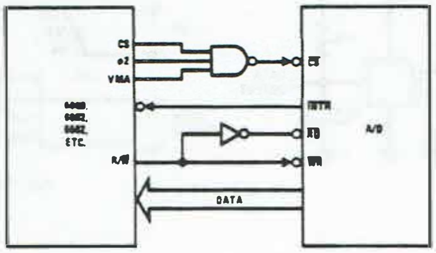


Note: Read strobe must occur 8 clock periods ($8/f_{CLK}$) after assertion of interrupt to guarantee reset of $\overline{INTR}$.

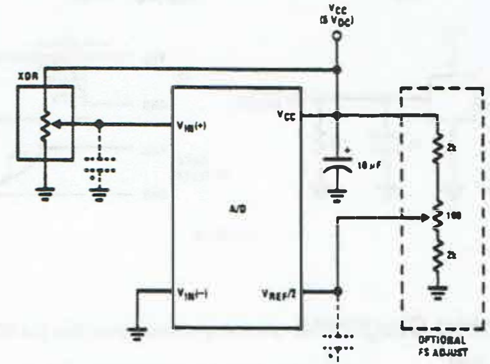
TL/H/5671-4

Typical Applications (Continued)

6800 Interface

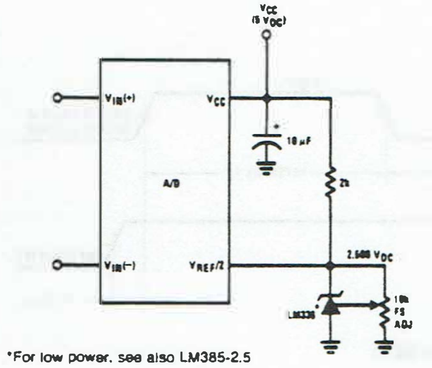


Ratiometric with Full-Scale Adjust



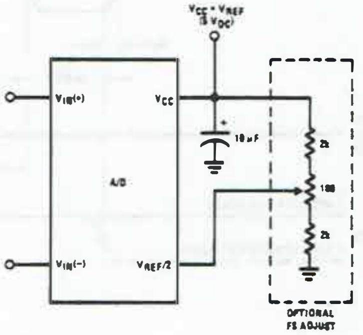
Note: before using caps at V_{IN} or $V_{REF}/2$, see section 2.3.2 Input Bypass Capacitors.

Absolute with a 2.500V Reference

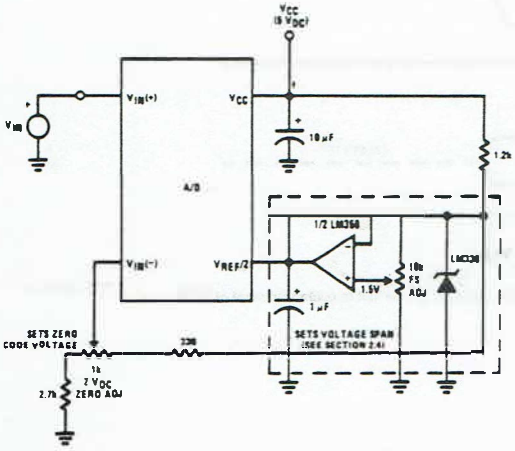


*For low power, see also LM385-2.5

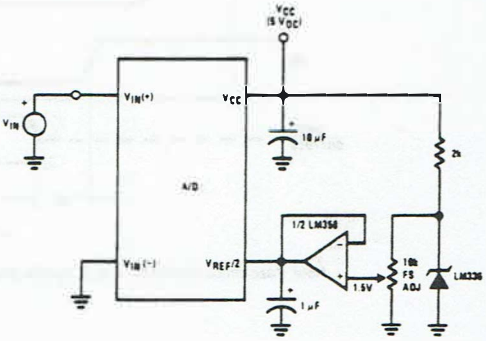
Absolute with a 5V Reference



Zero-Shift and Span Adjust: $2V \leq V_{IN} \leq 5V$



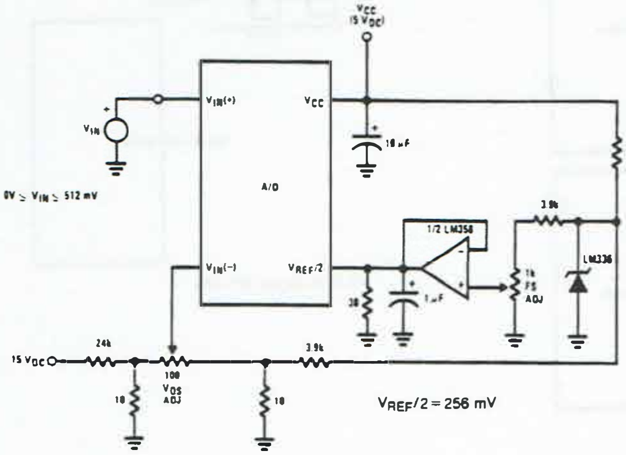
Span Adjust: $0V \leq V_{IN} \leq 3V$



TL/H/5671-5

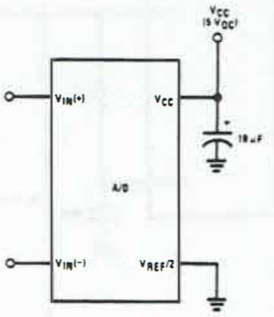
Typical Applications (Continued)

Directly Converting a Low-Level Signal



$V_{REF}/2 = 256$ mV

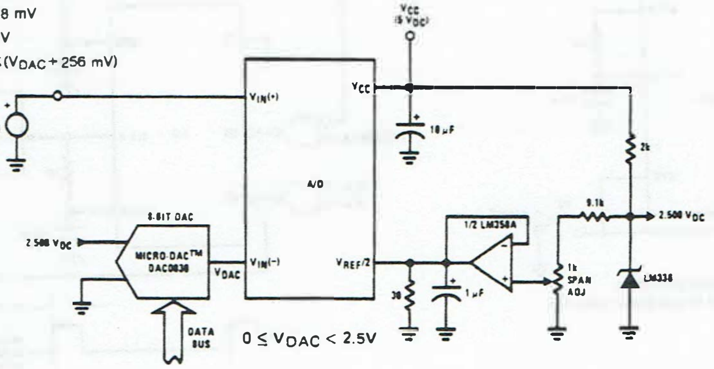
A µP Interfaced Comparator



For: $V_{IN}(+) > V_{IN}(-)$
Output = FF_{HEX}
For: $V_{IN}(+) < V_{IN}(-)$
Output = 00_{HEX}

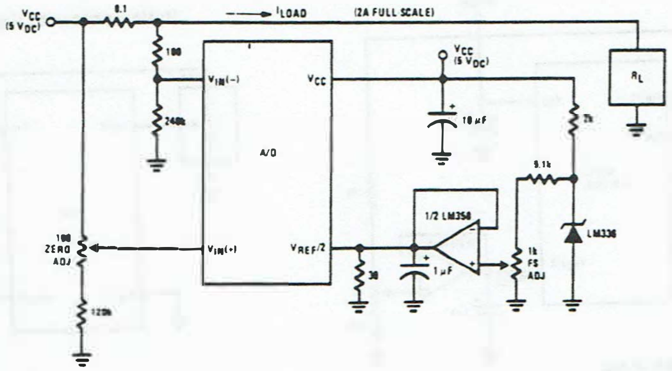
1 mV Resolution with µP Controlled Range

$V_{REF}/2 = 128$ mV
1 LSB = 1 mV
$V_{DAC} \leq V_{IN} \leq (V_{DAC} + 256$ mV)



$0 \leq V_{DAC} < 2.5$V

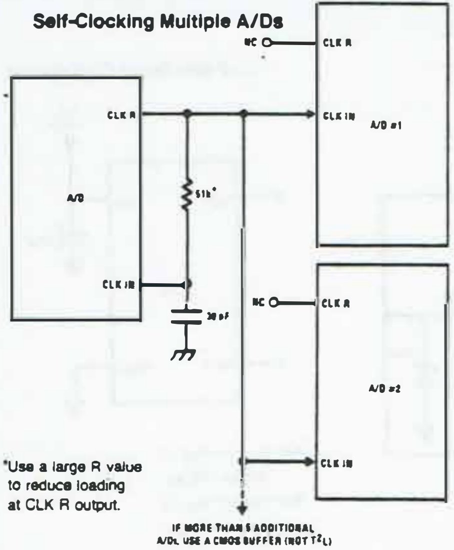
Digitizing a Current Flow



TL/H/5671–6

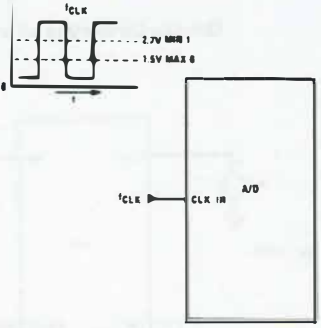
Typical Applications (Continued)

Self-Clocking Multiple A/Ds



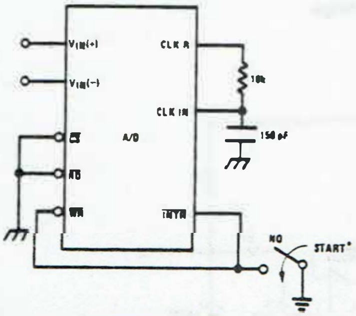
*Use a large R value to reduce loading at CLK R output.

External Clocking



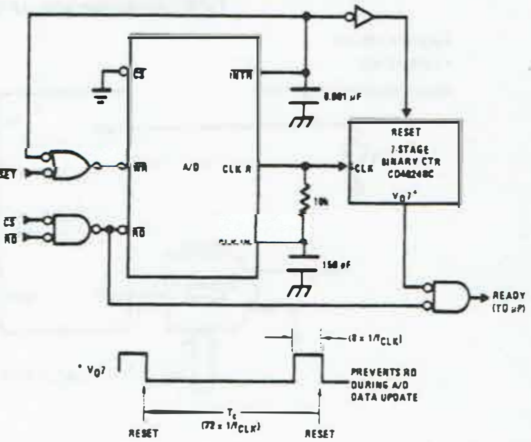
$100\text{ kHz} \le f_{CLK} \le 1460\text{ kHz}$

Self-Clocking in Free-Running Mode

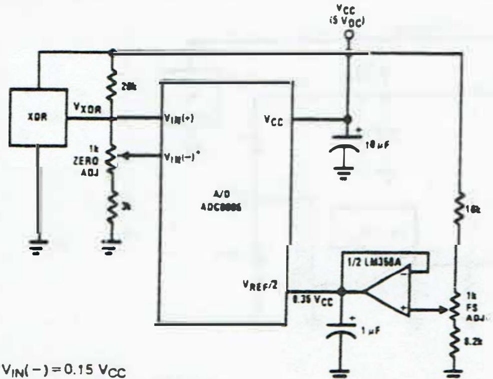


*After power-up, a momentary grounding of the $\overline{WR}$ input is needed to guarantee operation.

µP Interface for Free-Running A/D

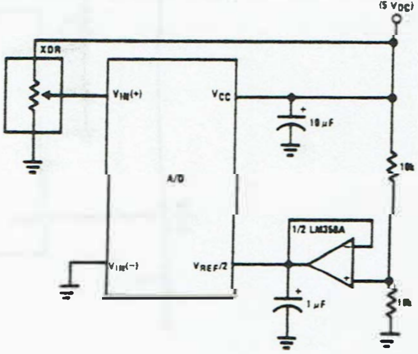


Operating with "Automotive" Ratiometric Transducers



$*V_{IN}(-) = 0.15\ V_{CC}$
$15\%\text{ of }V_{CC} \le V_{XDR} \le 85\%\text{ of }V_{CC}$

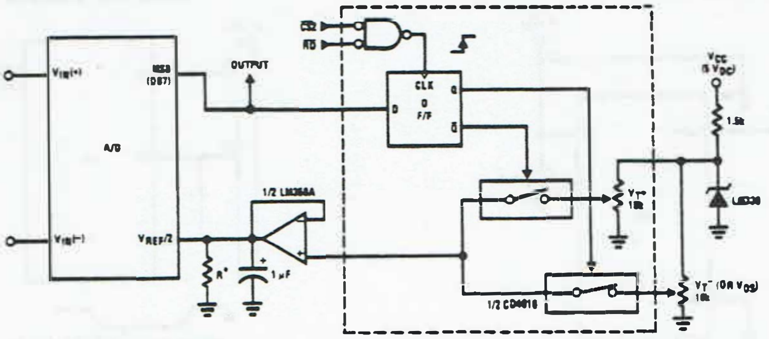
Ratiometric with $V_{REF}/2$ Forced



TL/H/5671-7

Typical Applications (Continued)

μP Compatible Differential-Input Comparator with Pre-Set V_{OS} (with or without Hysteresis)

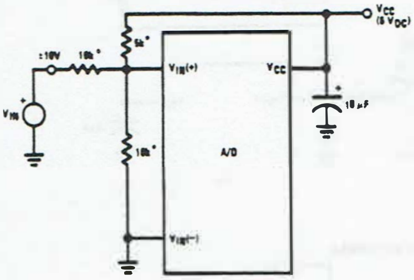


*See *Figure 5* to select R value

DB7 = "1" for $V_{IN}(+) > V_{IN}(-) + (V_{REF}/2)$

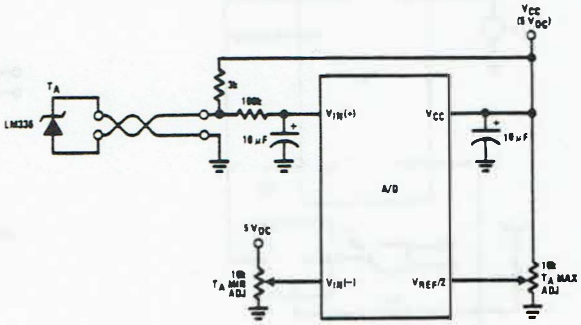
Omit circuitry within the dotted area if hysteresis is not needed

Handling ± 10V Analog Inputs

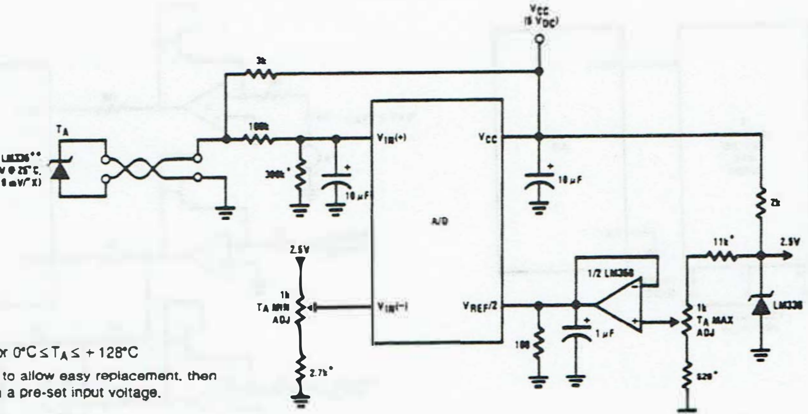


*Beckman Instruments #694-3-R10K resistor array

Low-Cost, μP Interfaced, Temperature-to-Digital Converter



μP Interfaced Temperature-to-Digital Converter



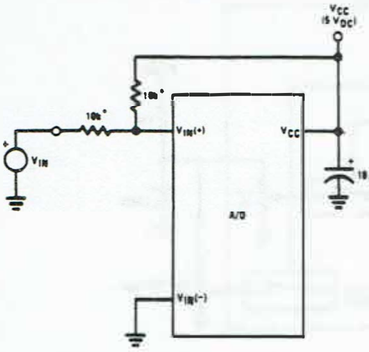
*Circuit values shown are for 0°C ≤ T_A ≤ + 128°C

**Can calibrate each sensor to allow easy replacement, then A/D can be calibrated with a pre-set input voltage.

TL/H/5671-8

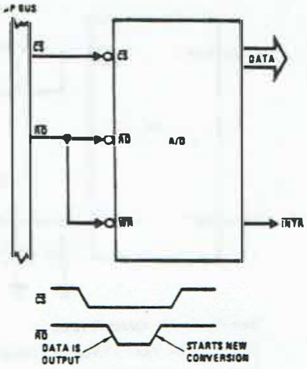
Typical Applications (Continued)

Handling ±5V Analog Inputs

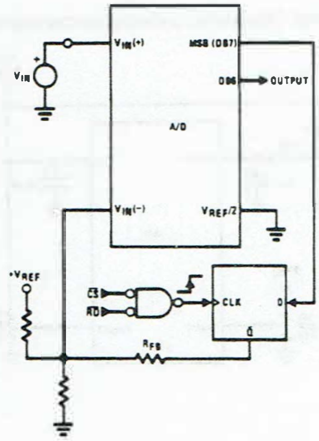


*Beckman Instruments #694-3-R10K resistor array

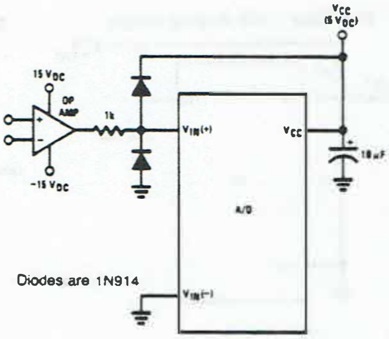
Read-Only Interface



µP Interfaced Comparator with Hysteresis

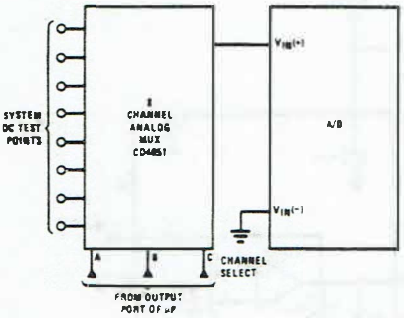


Protecting the Input

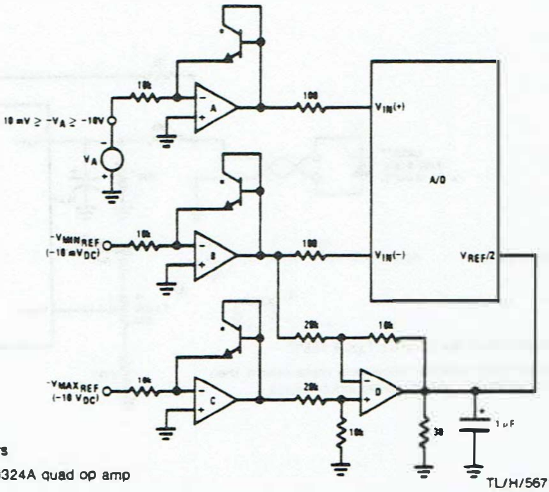


Diodes are 1N914

Analog Self-Test for a System



A Low-Cost, 3-Decade Logarithmic Converter



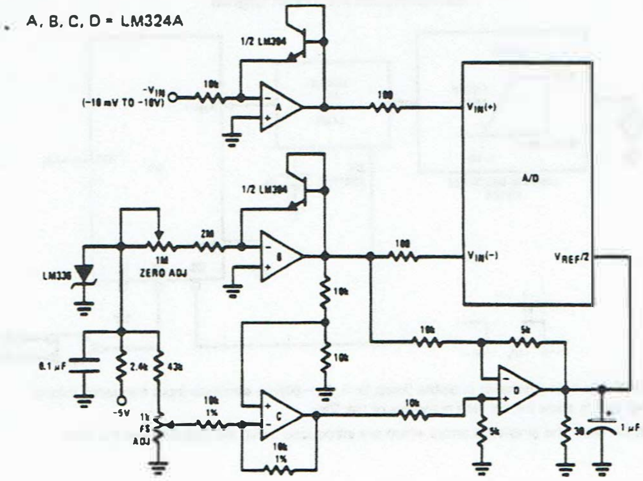
*LM389 transistors

A, B, C, D = LM324A quad op amp

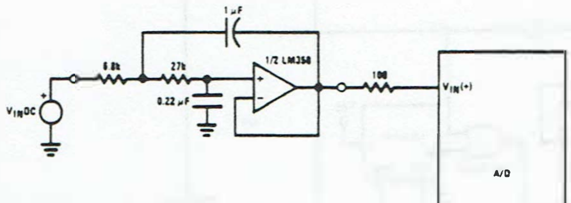
TL/H/5671-9

Typical Applications (Continued)

3-Decade Logarithmic A/D Converter



Noise Filtering the Analog Input

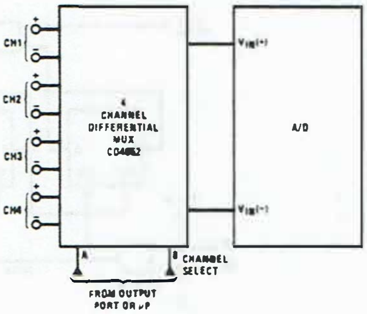


$f_C = 20$ Hz

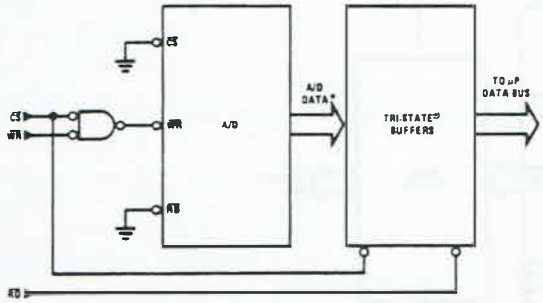
Uses Chebyshev implementation for steeper roll-off unity-gain, 2nd order, low-pass filter

Adding a separate filter for each channel increases system response time if an analog multiplexer is used

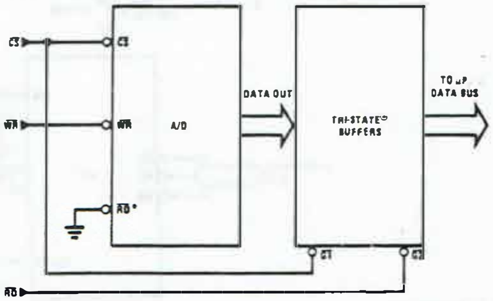
Multiplexing Differential Inputs



Output Buffers with A/D Data Enabled



Increasing Bus Drive and/or Reducing Time on Bus



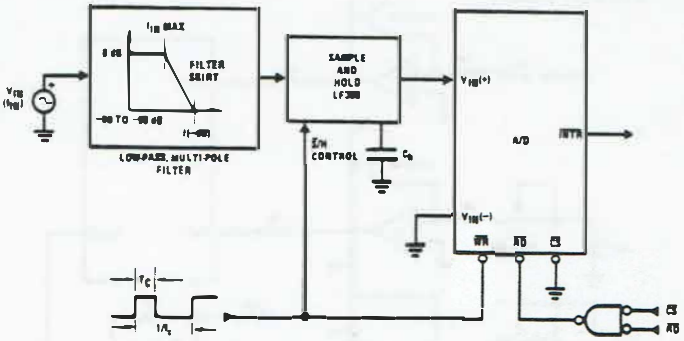
TL/H/5671-10

*A/D output data is updated 1 CLK period prior to assertion of $\overline{\text{INTR}}$

*Allows output data to set-up at falling edge of $\overline{\text{CS}}$

Typical Applications (Continued)

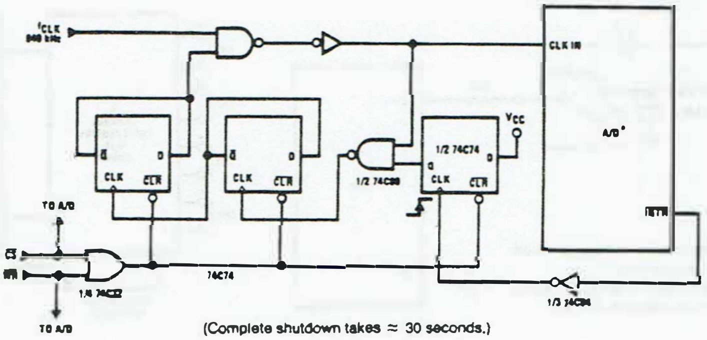
Sampling an AC Input Signal



Note 1: Oversample whenever possible [keep fs > 2f(−60)] to eliminate input frequency folding (aliasing) and to allow for the skirt response of the filter.

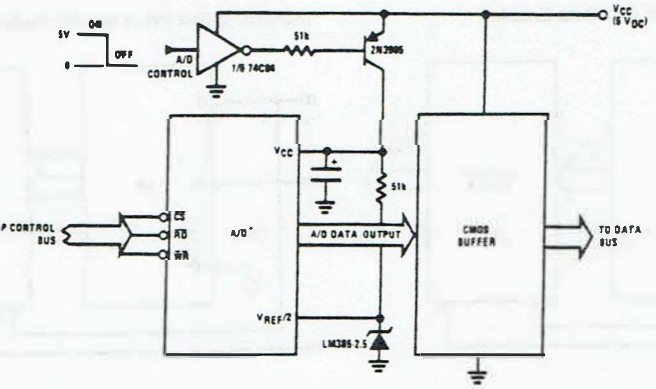
Note 2: Consider the amplitude errors which are introduced within the passband of the filter.

70% Power Savings by Clock Gating



(Complete shutdown takes ≈ 30 seconds.)

Power Savings by A/D and V_{REF} Shutdown



TL/H/5671-11

*Use ADC0801, 02, 03 or 05 for lowest power consumption.

Note: Logic inputs can be driven to V_{CC} with A/D supply at zero volts.

Buffer prevents data bus from overdriving output of A/D when in shutdown mode.

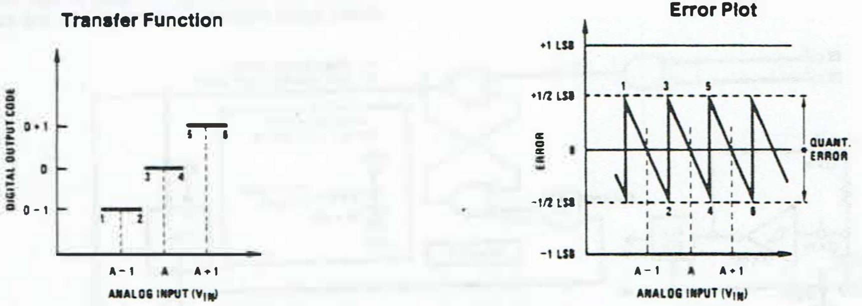
1.0 UNDERSTANDING A/D ERROR SPECS

A perfect A/D transfer characteristic (staircase waveform) is shown in *Figure 1a*. The horizontal scale is analog input voltage and the particular points labeled are in steps of 1 LSB (19.53 mV with 2.5V tied to the $V_{REF}/2$ pin). The digital output codes which correspond to these inputs are shown as D−1, D, and D+1. For the perfect A/D, not only will center-value (A−1, A, A+1, ...) analog inputs produce the correct output ditigal codes, but also each riser (the transitions between adjacent output codes) will be located ± ½ LSB away from each center-value. As shown, the risers are ideal and have no width. Correct digital output codes will be provided for a range of analog input voltages which extend ± ½ LSB from the ideal center-values. Each tread (the range of analog input voltage which provides the same digital output code) is therefore 1 LSB wide.

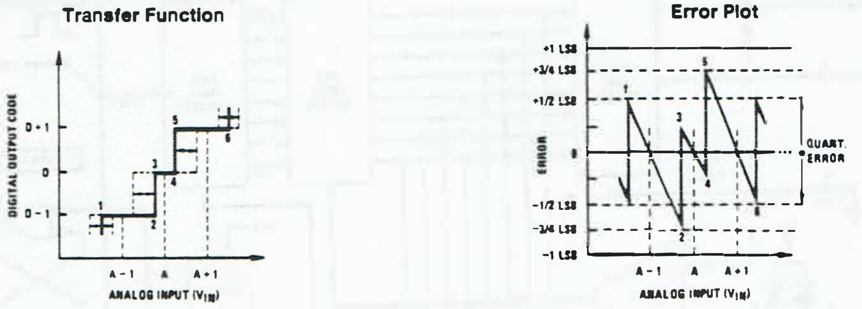
Figure 1b shows a worst case error plot for the ADC0801. All center-valued inputs are guaranteed to produce the correct output codes and the adjacent risers are guaranteed to be no closer to the center-value points than ± ¼ LSB. In other words, if we apply an analog input equal to the center-value ± ¼ LSB, *we guarantee* that the A/D will produce the correct digital code. The maximum range of the position of the code transition is indicated by the horizontal arrow and it is guaranteed to be no more than ½ LSB.

The error curve of *Figure 1c* shows a worst case error plot for the ADC0802. Here we guarantee that if we apply an analog input equal to the LSB analog voltage center-value the A/D will produce the correct digital code.

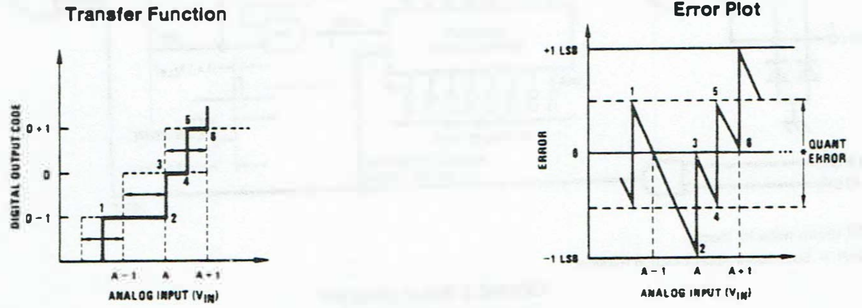
Next to each transfer function is shown the corresponding error plot. Many people may be more familiar with error plots than transfer functions. The analog input voltage to the A/D is provided by either a linear ramp or by the discrete output steps of a high resolution DAC. Notice that the error is continuously displayed and includes the quantization uncertainty of the A/D. For example the error at point 1 of *Figure 1a* is + ½ LSB because the digital code appeared ½ LSB in advance of the center-value of the tread. The error plots always have a constant negative slope and the abrupt upside steps are always 1 LSB in magnitude.



a) Accuracy = ±0 LSB A Perfect A/D



b) Accuracy = ± ¼ LSB



c) Accuracy = ± ½ LSB

TL/H/5671-12

FIGURE 1. Clarifying the Error Specs of an A/D Converter

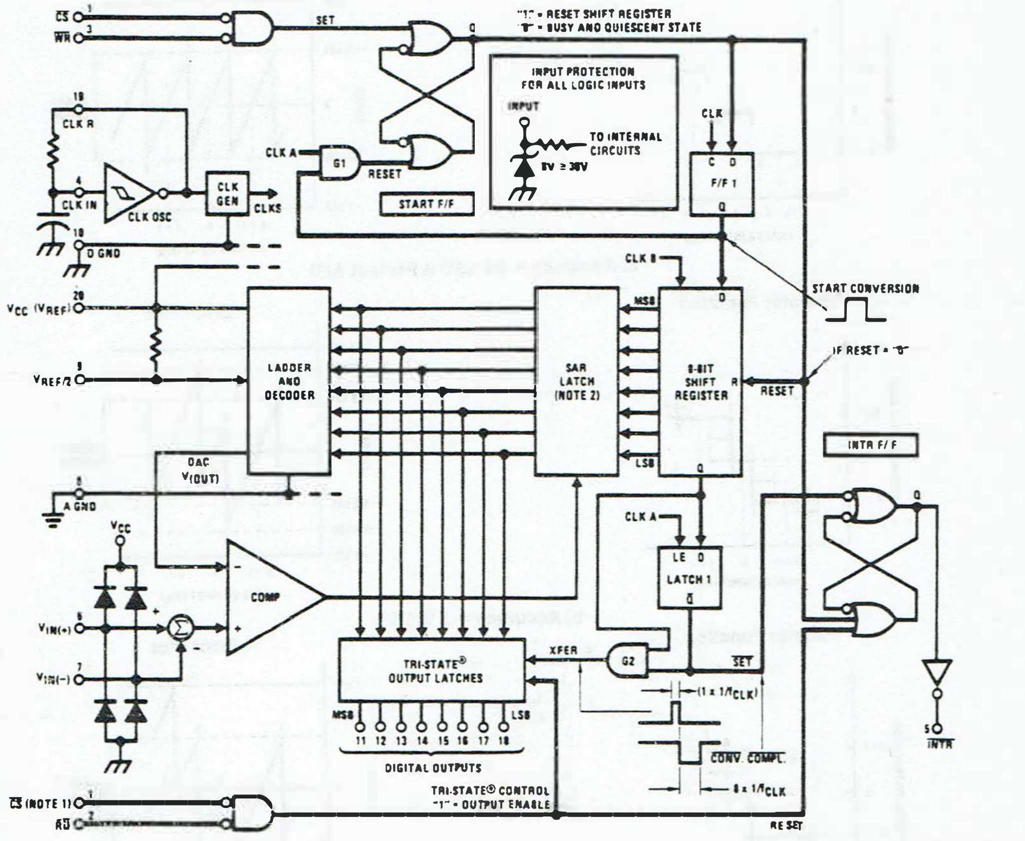
2.0 FUNCTIONAL DESCRIPTION

The ADC0801 series contains a circuit equivalent of the 256R network. Analog switches are sequenced by successive approximation logic to match the analog difference input voltage [$V_{IN}(+) - V_{IN}(-)$] to a corresponding tap on the R network. The most significant bit is tested first and after 8 comparisons (64 clock cycles) a digital 8-bit binary code (1111 1111 = full-scale) is transferred to an output latch and then an interrupt is asserted ($\overline{\text{INTR}}$ makes a high-to-low transition). A conversion in process can be interrupted by issuing a second start command. The device may be operated in the free-running mode by connecting $\overline{\text{INTR}}$ to the $\overline{\text{WR}}$ input with $\overline{\text{CS}} = 0$. To insure start-up under all possible conditions, an external $\overline{\text{WR}}$ pulse is required during the first power-up cycle.

On the high-to-low transition of the $\overline{\text{WR}}$ input the internal SAR latches and the shift register stages are reset. As long as the $\overline{\text{CS}}$ input and $\overline{\text{WR}}$ input remain low, the A/D will remain in a reset state. *Conversion will start from 1 to 8 clock periods after at least one of these inputs makes a low-to-high transition.*

A functional diagram of the A/D converter is shown in *Figure 2*. All of the package pinouts are shown and the major logic control paths are drawn in heavier weight lines.

The converter is started by having $\overline{\text{CS}}$ and $\overline{\text{WR}}$ simultaneously low. This sets the start flip-flop (F/F) and the resulting "1" level resets the 8-bit shift register, resets the Interrupt (INTR) F/F and inputs a "1" to the D flop, F/F1, which is at the input end of the 8-bit shift register. Internal clock signals then transfer this "1" to the Q output of F/F1. The AND gate, G1, combines this "1" output with a clock signal to provide a reset signal to the start F/F. If the set signal is no longer present (either $\overline{\text{WR}}$ or $\overline{\text{CS}}$ is a "1") the start F/F is reset and the 8-bit shift register then can have the "1" clocked in, which starts the conversion process. If the set signal were to still be present, this reset pulse would have no effect (both outputs of the start F/F would momentarily be at a "1" level) and the 8-bit shift register would continue to be held in the reset mode. This logic therefore allows for wide $\overline{\text{CS}}$ and $\overline{\text{WR}}$ signals and the converter will start after at least one of these signals returns high and the internal clocks again provide a reset signal for the start F/F.



TL/H/5671-13

Note 1: $\overline{\text{CS}}$ shown twice for clarity.

Note 2: SAR = Successive Approximation Register.

FIGURE 2. Block Diagram

After the "1" is clocked through the 8-bit shift register (which completes the SAR search) it appears as the input to the D-type latch, LATCH 1. As soon as this "1" is output from the shift register, the AND gate, G2, causes the new digital word to transfer to the TRI-STATE output latches. When LATCH 1 is subsequently enabled, the Q output makes a high-to-low transition which causes the INTR F/F to set. An inverting buffer then supplies the $\overline{\text{INTR}}$ input signal.

Note that this $\overline{\text{SET}}$ control of the INTR F/F remains low for 8 of the external clock periods (as the internal clocks run at ⅛ of the frequency of the external clock). If the data output is continuously enabled ($\overline{\text{CS}}$ and $\overline{\text{RD}}$ both held low), the $\overline{\text{INTR}}$ output will still signal the end of conversion (by a high-to-low transition), because the $\overline{\text{SET}}$ input can control the Q output of the INTR F/F even though the RESET input is constantly at a "1" level in this operating mode. This $\overline{\text{INTR}}$ output will therefore stay low for the duration of the $\overline{\text{SET}}$ signal, which is 8 periods of the external clock frequency (assuming the A/D is not started during this interval).

When operating in the free-running or continuous conversion mode ($\overline{\text{INTR}}$ pin tied to $\overline{\text{WR}}$ and $\overline{\text{CS}}$ wired low—see also section 2.8), the START F/F is SET by the high-to-low transition of the $\overline{\text{INTR}}$ signal. This resets the SHIFT REGISTER which causes the input to the D-type latch, LATCH 1, to go low. As the latch enable input is still present, the $\overline{\text{Q}}$ output will go high, which then allows the INTR F/F to be RESET. This reduces the width of the resulting $\overline{\text{INTR}}$ output pulse to only a few propagation delays (approximately 300 ns).

When data is to be read, the combination of both $\overline{\text{CS}}$ and $\overline{\text{RD}}$ being low will cause the INTR F/F to be reset and the TRI-STATE output latches will be enabled to provide the 8-bit digital outputs.

2.1 Digital Control Inputs

The digital control inputs ($\overline{\text{CS}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$) meet standard T^{2}L logic voltage levels. These signals have been renamed when compared to the standard A/D Start and Output Enable labels. In addition, these inputs are active low to allow an easy interface to microprocessor control busses. For non-microprocessor based applications, the $\overline{\text{CS}}$ input (pin 1) can be grounded and the standard A/D Start function is obtained by an active low pulse applied at the $\overline{\text{WR}}$ input (pin 3) and the Output Enable function is caused by an active low pulse at the $\overline{\text{RD}}$ input (pin 2).

2.2 Analog Differential Voltage Inputs and Common-Mode Rejection

This A/D has additional applications flexibility due to the analog differential voltage input. The $V_{IN}(-)$ input (pin 7) can be used to automatically subtract a fixed voltage value from the input reading (tare correction). This is also useful in 4 mA–20 mA current loop conversion. In addition, common-mode noise can be reduced by use of the differential input.

The time interval between sampling $V_{IN}(+)$ and $V_{IN}(-)$ is 4-½ clock periods. The maximum error voltage due to this slight time difference between the input voltage samples is given by:

$$\Delta V_e(MAX) = (V_P)(2\pi f_{cm})\left(\frac{4.5}{f_{CLK}}\right),$$

where:

ΔV_e is the error voltage due to sampling delay

V_P is the peak value of the common-mode voltage

f_{cm} is the common-mode frequency

As an example, to keep this error to ¼ LSB (~5 mV) when operating with a 60 Hz common-mode frequency, f_{cm}, and using a 640 kHz A/D clock, f_{CLK}, would allow a peak value of the common-mode voltage, V_P, which is given by:

$$V_P = \frac{[\Delta V_{e(MAX)}(f_{CLK})]}{(2\pi f_{cm})(4.5)}$$

or

$$V_P = \frac{(5 \times 10^{-3})(640 \times 10^3)}{(6.28)(60)(4.5)}$$

which gives

$$V_P \cong 1.9V.$$

The allowed range of analog input voltages usually places more severe restrictions on input common-mode noise levels.

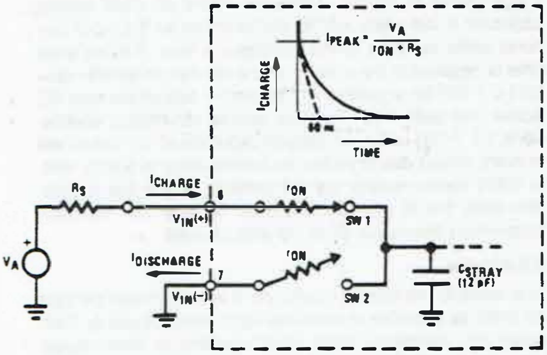
An analog input voltage with a reduced span and a relatively large zero offset can be easily handled by making use of the differential input (see section 2.4 Reference Voltage).

2.3 Analog Inputs

2.3.1 Input Current

Normal Mode

Due to the internal switching action, displacement currents will flow at the analog inputs. This is due to on-chip stray capacitance to ground as shown in *Figure 3*.



TL/H/5671–14

r_{ON} of SW 1 and SW 2 ≅ 5 kΩ

$r = r_{ON}\, C_{STRAY} \cong 5\ k\Omega \times 12\ pF = 60\ ns$

FIGURE 3. Analog Input Impedance

The voltage on this capacitance is switched and will result in currents entering the $V_{IN}(+)$ input pin and leaving the $V_{IN}(-)$ input which will depend on the analog differential input voltage levels. These current transients occur at the leading edge of the internal clocks. They rapidly decay and *do not cause errors* as the on-chip comparator is strobed at the end of the clock period.

Fault Mode

If the voltage source which is applied to the $V_{IN}(+)$ pin exceeds the allowed operating range of $V_{CC} + 50$ mV, large input currents can flow through a parasitic diode to the V_{CC} pin. If these currents could exceed the 1 mA max allowed spec, an external diode (1N914) should be added to bypass this current to the V_{CC} pin (with the current bypassed with this diode, the voltage at the $V_{IN}(+)$ pin can exceed the V_{CC} voltage by the forward voltage of this diode).

2.3.2 Input Bypass Capacitors

Bypass capacitors at the inputs will average these charges and cause a DC current to flow through the output resistances of the analog signal sources. This charge pumping action is worse for continuous conversions with the $V_{IN}(+)$ input voltage at full-scale. For continuous conversions with a 640 kHz clock frequency with the $V_{IN}(+)$ input at 5V, this DC current is at a maximum of approximately 5 μA. Therefore, *bypass capacitors should not be used at the analog inputs or the $V_{REF}/2$ pin* for high resistance sources (> 1 kΩ). If input bypass capacitors are necessary for noise filtering and high source resistance is desirable to minimize capacitor size, the detrimental effects of the voltage drop across this input resistance, which is due to the average value of the input current, can be eliminated with a full-scale adjustment while the given source resistor and input bypass capacitor are both in place. This is possible because the average value of the input current is a precise linear function of the differential input voltage.

2.3.3 Input Source Resistance

Large values of source resistance where an input bypass capacitor is not used, *will not cause errors* as the input currents settle out prior to the comparison time. If a low pass filter is required in the system, use a low valued series resistor ($\leq$ 1 kΩ) for a passive RC section or add an op amp RC active low pass filter. For low source resistance applications, ($\leq$ 1 kΩ), a 0.1 μF bypass capacitor at the inputs will prevent pickup due to series lead inductance of a long wire. A 100Ω series resistor can be used to isolate this capacitor—both the R and C are placed outside the feedback loop—from the output of an op amp, if used.

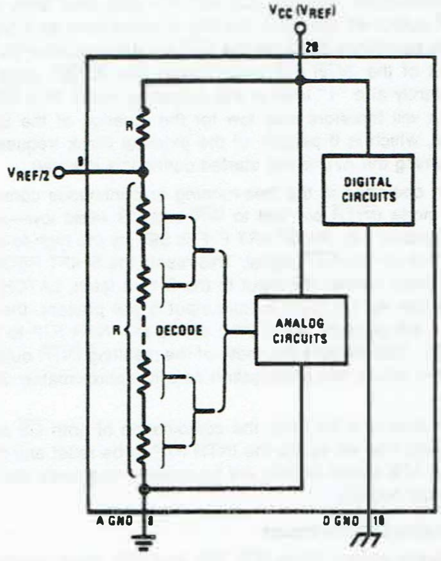
2.3.4 Noise

The leads to the analog inputs (pin 6 and 7) should be kept as short as possible to minimize input noise coupling. Both noise and undesired digital clock coupling to these inputs can cause system errors. The source resistance for these inputs should, in general, be kept below 5 kΩ. Larger values of source resistance can cause undesired system noise pickup. Input bypass capacitors, placed from the analog inputs to ground, will eliminate system noise pickup but can create analog scale errors as these capacitors will average the transient input switching currents of the A/D (see section 2.3.1.). This scale error depends on both a large source resistance and the use of an input bypass capacitor. This error can be eliminated by doing a full-scale adjustment of the A/D (adjust $V_{REF}/2$ for a proper full-scale reading—see section 2.5.2 on Full-Scale Adjustment) with the source resistance and input bypass capacitor in place.

2.4 Reference Voltage

2.4.1 Span Adjust

For maximum applications flexibility, these A/Ds have been designed to accommodate a 5 V_{DC}, 2.5 V_{DC} or an adjusted voltage reference. This has been achieved in the design of the IC as shown in *Figure 4*.

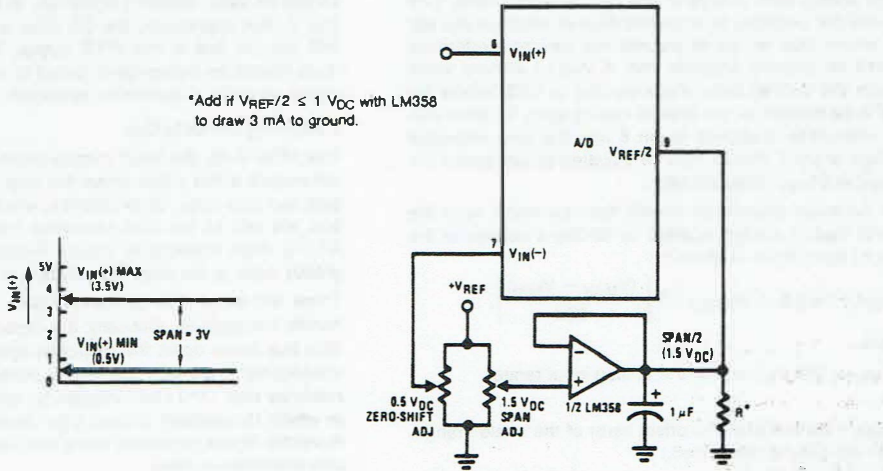


TL/H/5671-15

FIGURE 4. The $V_{REFERENCE}$ Design on the IC

Notice that the reference voltage for the IC is either ½ of the voltage which is applied to the V_{CC} supply pin, or is equal to the voltage which is externally forced at the $V_{REF}/2$ pin. This allows for a ratiometric voltage reference using the V_{CC} supply, a 5 V_{DC} reference voltage can be used for the V_{CC} supply or a voltage less than 2.5 V_{DC} can be applied to the $V_{REF}/2$ input for increased application flexibility. The internal gain to the $V_{REF}/2$ input is 2 making the full-scale differential input voltage twice the voltage at pin 9.

An example of the use of an adjusted reference voltage is to accommodate a reduced span—or dynamic voltage range of the analog input voltage. If the analog input voltage were to range from 0.5 V_{DC} to 3.5 V_{DC}, instead of 0V to 5 V_{DC}, the span would be 3V as shown in *Figure 5*. With 0.5 V_{DC} applied to the $V_{IN}(-)$ pin to absorb the offset, the reference voltage can be made equal to ½ of the 3V span or 1.5 V_{DC}. The A/D now will encode the $V_{IN}(+)$ signal from 0.5V to 3.5 V with the 0.5V input corresponding to zero and the 3.5 V_{DC} input corresponding to full-scale. The full 8 bits of resolution are therefore applied over this reduced analog input voltage range.



*Add if $V_{REF}/2 \leq 1\ V_{DC}$ with LM358 to draw 3 mA to ground.

a) Analog Input Signal Example

b) Accommodating an Analog Input from 0.5V (Digital Out = = 00_{HEX}) to 3.5V (Digital Out = FF_{HEX})

FIGURE 5. Adapting the A/D Analog Input Voltages to Match an Arbitrary Input Signal Range

2.4.2 Reference Accuracy Requirements

The converter can be operated in a ratiometric mode or an absolute mode. In ratiometric converter applications, the magnitude of the reference voltage is a factor in both the output of the source transducer and the output of the A/D converter and therefore cancels out in the final digital output code. The ADC0805 is specified particularly for use in ratiometric applications with no adjustments required. In absolute conversion applications, both the initial value and the temperature stability of the reference voltage are important accuracy factors in the operation of the A/D converter. For $V_{REF}/2$ voltages of 2.4 V_{DC} nominal value, initial errors of ±10 mV_{DC} will cause conversion errors of ±1 LSB due to the gain of 2 of the $V_{REF}/2$ input. In reduced span applications, the initial value and the stability of the $V_{REF}/2$ input voltage become even more important. For example, if the span is reduced to 2.5V, the analog input LSB voltage value is correspondingly reduced from 20 mV (5V span) to 10 mV and 1 LSB at the $V_{REF}/2$ input becomes 5 mV. As can be seen, this reduces the allowed initial tolerance of the reference voltage and requires correspondingly less absolute change with temperature variations. Note that spans smaller than 2.5V place even tighter requirements on the initial accuracy and stability of the reference source.

In general, the magnitude of the reference voltage will require an initial adjustment. Errors due to an improper value of reference voltage appear as full-scale errors in the A/D transfer function. IC voltage regulators may be used for references if the ambient temperature changes are not excessive. The LM336B 2.5V IC reference diode (from National Semiconductor) is available which has a temperature stability of 1.8 mV typ (6 mV max) over $0°C \leq T_A \leq +70°C$. Other temperature range parts are also available.

2.5 Errors and Reference Voltage Adjustments

2.5.1 Zero Error

The zero of the A/D does not require adjustment. If the minimum analog input voltage value, $V_{IN(MIN)}$, is not ground, a zero offset can be done. The converter can be made to output 0000 0000 digital code for this minimum input voltage by biasing the A/D $V_{IN}(-)$ input at this $V_{IN(MIN)}$ value (see Applications section). This utilizes the differential mode operation of the A/D.

The zero error of the A/D converter relates to the location of the first riser of the transfer function and can be measured by grounding the V (−) input and applying a small magnitude positive voltage to the V (+) input. Zero error is the difference between the actual DC input voltage which is necessary to just cause an output digital code transition from 0000 0000 to 0000 0001 and the ideal ½ LSB value (½ LSB = 9.8 mV for $V_{REF}/2 = 2.500\ V_{DC}$).

2.5.2 Full-Scale

The full-scale adjustment can be made by applying a differential input voltage which is 1½ LSB down from the desired analog full-scale voltage range and then adjusting the magnitude of the $V_{REF}/2$ input (pin 9 or the V_{CC} supply if pin 9 is not used) for a digital output code which is just changing from 1111 1110 to 1111 1111.

2.5.3 Adjusting for an Arbitrary Analog Input Voltage Range

If the analog zero voltage of the A/D is shifted away from ground (for example, to accommodate an analog input signal which does not go to ground) this new zero reference should be properly adjusted first. A $V_{IN}(+)$ voltage which equals this desired zero reference plus ½ LSB (where the LSB is calculated for the desired analog span, 1 LSB = analog span/256) is applied to pin 6 and the zero reference voltage at pin 7 should then be adjusted to just obtain the 00_{HEX} to 01_{HEX} code transition.

The full-scale adjustment should then be made (with the proper $V_{IN}(-)$ voltage applied) by forcing a voltage to the $V_{IN}(+)$ input which is given by:

$$V_{IN}(+)\text{ fs adj} = V_{MAX} - 1.5\left[\frac{(V_{MAX} - V_{MIN})}{256}\right],$$

where:

V_{MAX} = The high end of the analog input range

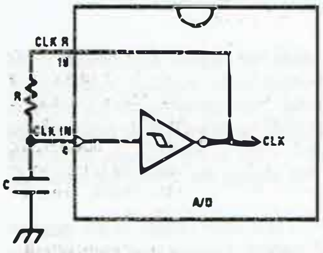
and

V_{MIN} = the low end (the offset zero) of the analog range. (Both are ground referenced.)

The $V_{REF}/2$ (or V_{CC}) voltage is then adjusted to provide a code change from FE_{HEX} to FF_{HEX}. This completes the adjustment procedure.

2.6 Clocking Option

The clock for the A/D can be derived from the CPU clock or an external RC can be added to provide self-clocking. The CLK IN (pin 4) makes use of a Schmitt trigger as shown in *Figure 6*.



$$f_{CLK} \cong \frac{1}{1.1\,RC}$$

$$R \cong 10\text{ k}\Omega$$

TL/H/5671-17

FIGURE 6. Self-Clocking the A/D

Heavy capacitive or DC loading of the clock R pin should be avoided as this will disturb normal converter operation. Loads less than 50 pF, such as driving up to 7 A/D converter clock inputs from a single clock R pin of 1 converter, are allowed. For larger clock line loading, a CMOS or low power T^2L buffer or PNP input logic should be used to minimize the loading on the clock R pin (do not use a standard T^2L buffer).

2.7 Restart During a Conversion

If the A/D is restarted ($\overline{CS}$ and $\overline{WR}$ go low and return high) during a conversion, the converter is reset and a new conversion is started. The output data latch is not updated if the conversion in process is not allowed to be completed, therefore the data of the previous conversion remains in this latch. The $\overline{INTR}$ output also simple remains at the "1" level.

2.8 Continuous Conversions

For operation in the free-running mode an initializing pulse should be used, following power-up, to insure circuit operation. In this application, the $\overline{CS}$ input is grounded and the $\overline{WR}$ input is tied to the $\overline{INTR}$ output. This $\overline{WR}$ and $\overline{INTR}$ node should be momentarily forced to logic low following a power-up cycle to guarantee operation.

2.9 Driving the Data Bus

This MOS A/D, like MOS microprocessors and memories, will require a bus driver when the total capacitance of the data bus gets large. Other circuitry, which is tied to the data bus, will add to the total capacitive loading, even in TRI-STATE (high impedance mode). Backplane bussing also greatly adds to the stray capacitance of the data bus.

There are some alternatives available to the designer to handle this problem. Basically, the capacitive loading of the data bus slows down the response time, even though DC specifications are still met. For systems operating with a relatively slow CPU clock frequency, more time is available in which to establish proper logic levels on the bus and therefore higher capacitive loads can be driven (see typical characteristics curves).

At higher CPU clock frequencies time can be extended for I/O reads (and/or writes) by inserting wait states (8080) or using clock extending circuits (6800).

Finally, if time is short and capacitive loading is high, external bus drivers must be used. These can be TRI-STATE buffers (low power Schottky is recommended such as the DM74LS240 series) or special higher drive current products which are designed as bus drivers. High current bipolar bus drivers with PNP inputs are recommended.

2.10 Power Supplies

Noise spikes on the V_{CC} supply line can cause conversion errors as the comparator will respond to this noise. A low inductance tantalum filter capacitor should be used close to the converter V_{CC} pin and values of 1 μF or greater are recommended. If an unregulated voltage is available in the system, a separate LM340LAZ-5.0, TO-92, 5V voltage regulator for the converter (and other analog circuitry) will greatly reduce digital noise on the V_{CC} supply.

2.11 Wiring and Hook-Up Precautions

Standard digital wire wrap sockets are not satisfactory for breadboarding this A/D converter. Sockets on PC boards can be used and all logic signal wires and leads should be grouped and kept as far away as possible from the analog signal leads. Exposed leads to the analog inputs can cause undesired digital noise and hum pickup, therefore shielded leads may be necessary in many applications.

A single point analog ground should be used which is separate from the logic ground points. The power supply bypass capacitor and the self-clocking capacitor (if used) should both be returned to digital ground. Any $V_{REF}/2$ bypass capacitors, analog input filter capacitors, or input signal shielding should be returned to the analog ground point. A test for proper grounding is to measure the zero error of the A/D converter. Zero errors in excess of ¼ LSB can usually be traced to improper board layout and wiring (see section 2.5.1 for measuring the zero error).

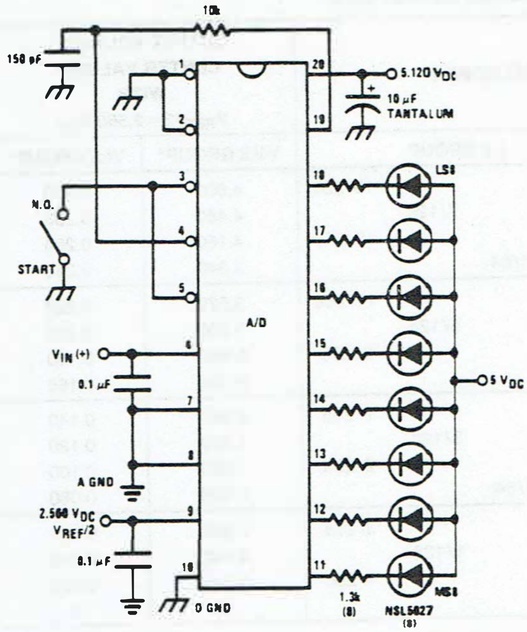
3.0 TESTING THE A/D CONVERTER

There are many degrees of complexity associated with testing an A/D converter. One of the simplest tests is to apply a known analog input voltage to the converter and use LEDs to display the resulting digital output code as shown in *Figure 7*.

For ease of testing, the $V_{REF}/2$ (pin 9) should be supplied with 2.560 V_{DC} and a V_{CC} supply voltage of 5.12 V_{DC} should be used. This provides an LSB value of 20 mV.

If a full-scale adjustment is to be made, an analog input voltage of 5.090 V_{DC} (5.120−1½ LSB) should be applied to the $V_{IN}(+)$ pin with the $V_{IN}(-)$ pin grounded. The value of the $V_{REF}/2$ input voltage should then be adjusted until the digital output code is just changing from 1111 1110 to 1111 1111. This value of $V_{REF}/2$ should then be used for all the tests.

The digital output LED display can be decoded by dividing the 8 bits into 2 hex characters, the 4 most significant (MS) and the 4 least significant (LS). Table I shows the fractional binary equivalent of these two 4-bit groups. By adding the decoded voltages which are obtained from the column: Input voltage value for a 2.560 $V_{REF}/2$ of both the MS and the LS groups, the value of the digital display can be determined. For example, for an output LED display of 1011 0110 or B6 (in hex), the voltage values from the table are 3.520 + 0.120 or 3.640 V_{DC}. These voltage values represent the center-values of a perfect A/D converter. The effects of quantization error have to be accounted for in the interpretation of the test results.



TL/H/5671-18

FIGURE 7. Basic A/D Tester

For a higher speed test system, or to obtain plotted data, a digital-to-analog converter is needed for the test set-up. An accurate 10-bit DAC can serve as the precision voltage source for the A/D. Errors of the A/D under test can be provided as either analog voltages or differences in 2 ditigal words.

A basic A/D tester which uses a DAC and provides the error as an analog output voltage is shown in *Figure 8*. The 2 op amps can be elminated if a lab DVM with a numerical subtraction feature is available to directly readout the difference voltage, "A−C". The analog input voltage can be supplied by a low frequency ramp generator and an X-Y plotter can be used to provide analog error (Y axis) versus analog input (X axis). The construction details of a tester of this type are provided in the NSC application note AN-179, "Analog-to-Digital Converter Testing".

For operation with a microprocessor or a computer-based test system, it is more convenient to present the errors digitally. This can be done with the circuit of *Figure 9*, where the output code transitions can be detected as the 10-bit DAC is incremented. This provides ¼ LSB steps for the 8-bit A/D under test. If the results of this test are automatically plotted with the analog input on the X axis and the error (in LSB's) as the Y axis, a useful transfer function of the A/D under test results. For acceptance testing, the plot is not necessary and the testing speed can be increased by establishing internal limits on the allowed error for each code.

4.0 MICROPROCESSOR INTERFACING

To dicuss the interface with 8080A and 6800 microprocessors, a common sample subroutine structure is used. The microprocessor starts the A/D, reads and stores the results of 16 successive conversions, then returns to the user's program. The 16 data bytes are stored in 16 successive memory locations. All Data and Addresses will be given in hexadecimal form. Software and hardware details are provided separately for each type of microprocessor.

4.1 Interfacing 8080 Microprocessor Derivatives (8048, 8085)

This converter has been designed to directly interface with derivatives of the 8080 microprocessor. The A/D can be mapped into memory space (using standard memory address decoding for $\overline{CS}$ and the $\overline{MEMR}$ and $\overline{MEMW}$ strobes) or it can be controlled as an I/O device by using the $\overline{I/O\ R}$ and $\overline{I/O\ W}$ strobes and decoding the address bits A0 → A7 (or address bits A8 → A15 as they will contain the same 8-bit address information) to obtain the $\overline{CS}$ input. Using the I/O space provides 256 additional addresses and may allow a simpler 8-bit address decoder but the data can only be input to the accumulator. To make use of the additional memory reference instructions, the A/D should be mapped into memory space. An example of an A/D in I/O space is shown in *Figure 10*.

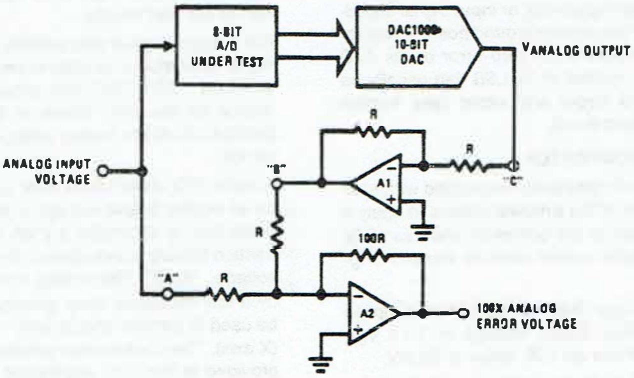
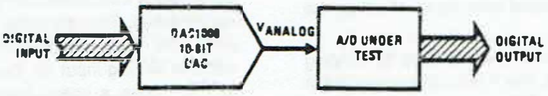


FIGURE 8. A/D Tester with Analog Error Output



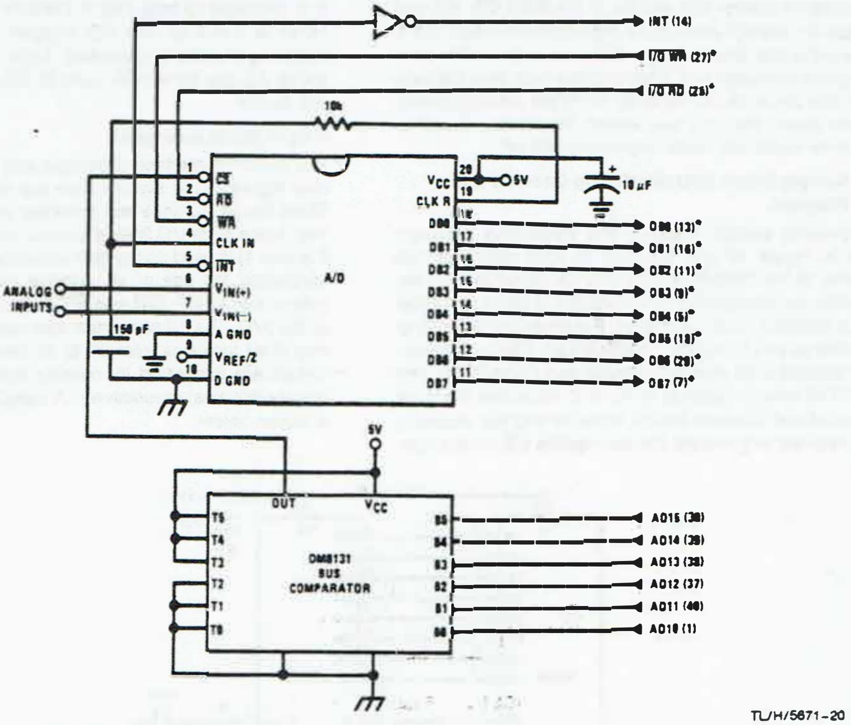
TL/H/5671-19

FIGURE 9. Basic "Digital" A/D Tester

TABLE I. DECODING THE DIGITAL OUTPUT LEDs

HEX	BINARY				FRACTIONAL BINARY VALUE FOR		OUTPUT VOLTAGE CENTER VALUES WITH $V_{REF}/2 = 2.560\ V_{DC}$	
					MS GROUP	LS GROUP	VMS GROUP*	VLS GROUP*
F	1	1	1	1	15/16	15/256	4.800	0.300
E	1	1	1	0	7/8	7/128	4.480	0.280
D	1	1	0	1	13/16	13/256	4.160	0.260
C	1	1	0	0	3/4	3/64	3.840	0.240
B	1	0	1	1	11/16	11/256	3.520	0.220
A	1	0	1	0	5/8	5/128	3.200	0.200
9	1	0	0	1	9/16	9/256	2/880	0.180
8	1	0	0	0	1/2	1/32	2/560	0.160
7	0	1	1	1	7/16	7/256	2.240	0.140
6	0	1	1	0	3/8	3/128	1.920	0.120
5	0	1	0	1	5/16	2/256	1.600	0.100
4	0	1	0	0	1/4	1/64	1/280	0.080
3	0	0	1	1	3/16	3/256	0.960	0.060
2	0	0	1	0	1/8	1/128	0.640	0.040
1	0	0	0	1	1/16	1/256	0.320	0.020
0	0	0	0	0			0	0

*Display Output = VMS Group + VLS Group



TL/H/5671-20

Note 1: *Pin numbers for the INS8228 system controller, others are INS8080A.

Note 2: Pin 23 of the INS8228 must be tied to +12V through a 1 kΩ resistor to generate the RST 7 instruction when an interrupt is acknowledged as required by the accompanying sample program.

FIGURE 10. ADC0801-INS8080A CPU Interface

SAMPLE PROGRAM FOR *FIGURE 10* ADC0801-INS8080A CPU INTERFACE

```
0038    C3 00 03   RST 7:            JMP   LD DATA
  •        •          •
  •        •          •
0100    21 00 02   START:            LXI H 0200H    ; HL pair will point to
                                                    ; data storage locations
0103    31 00 04   RETURN:           LXI SP 0400H   ; Initialize stack pointer (Note 1)
0106    7D                           MOV A, L       ; Test # of bytes entered
0107    FE 0F                        CPI 0F H       ; If # = 16. JMP to
0109    CA 13 01                     JZ CONT        ; user program
010C    D3 E0                        OUT E0 H       ; Start A/D
010E    FB                           EI             ; Enable interrupt
010F    00         LOOP:             NOP            ; Loop until end of
0110    C3 0F 01                     JMP LOOP       ; conversion
0113         •     CONT:               •
  •          •       •                 •
  •          •     (User program to    •
  •          •     process data)       •
  •          •       •                 •
  •          •       •                 •
0300    DB E0      LD DATA:          IN E0 H        ; Load data into accumulator
0302    77                           MOV M, A       ; Store data
0303    23                           INX H          ; Increment storage pointer
0304    C3 03 01                     JMP RETURN
```

Note 1: The stack pointer must be dimensioned because a RST 7 instruction pushes the PC onto the stack.

Note 2: All address used were arbitrarily chosen.

The standard control bus signals of the 8080 $\overline{CS}$, $\overline{RD}$ and $\overline{WR}$) can be directly wired to the digital control inputs of the A/D and the bus timing requirements are met to allow both starting the converter and outputting the data onto the data bus. A bus driver should be used for larger microprocessor systems where the data bus leaves the PC board and/or must drive capacitive loads larger than 100 pF.

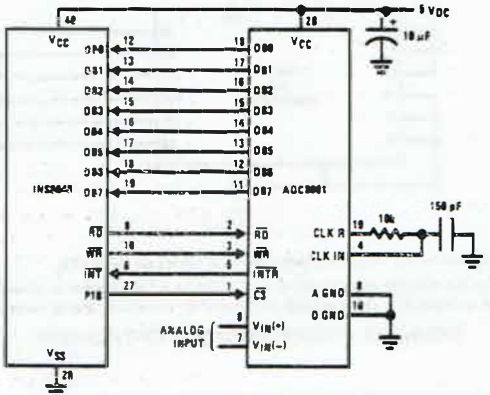
4.1.1 Sample 8080A CPU Interfacing Circuitry and Program

The following sample program and associated hardware shown in *Figure 10* may be used to input data from the converter to the INS8080A CPU chip set (comprised of the INS8080A microprocessor, the INS8228 system controller and the INS8224 clock generator). For simplicity, the A/D is controlled as an I/O device, specifically an 8-bit bi-directional port located at an arbitrarily chosen port address, E0. The TRI-STATE output capability of the A/D eliminates the need for a peripheral interface device, however address decoding is still required to generate the appropriate $\overline{CS}$ for the converter.

It is important to note that in systems where the A/D converter is 1-of-8 or less I/O mapped devices, no address decoding circuitry is necessary. Each of the 8 address bits (A0 to A7) can be directly used as $\overline{CS}$ inputs—one for each I/O device.

4.1.2 INS8048 Interface

The INS8048 interface technique with the ADC0801 series (see *Figure 11*) is simpler than the 8080A CPU interface. There are 24 I/O lines and three test input lines in the 8048. With these extra I/O lines available, one of the I/O lines (bit 0 of port 1) is used as the chip select signal to the A/D, thus eliminating the use of an external address decoder. Bus control signals $\overline{RD}$, $\overline{WR}$ and $\overline{INT}$ of the 8048 are tied directly to the A/D. The 16 converted data words are stored at on-chip RAM locations from 20 to 2F (Hex). The $\overline{RD}$ and $\overline{WR}$ signals are generated by reading from and writing into a dummy address, respectively. A sample interface program is shown below.

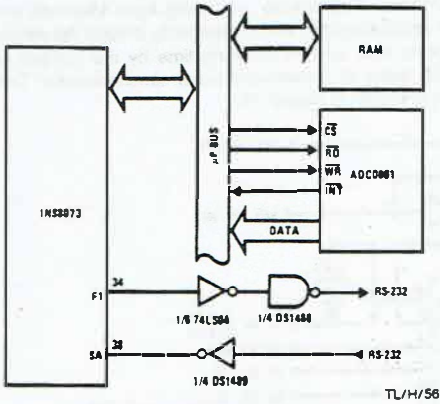


TL/H/5671–21

FIGURE 11. INS8048 Interface

SAMPLE PROGRAM FOR *FIGURE 11* INS8048 INTERFACE

```
04 10                 JMP      10H           ; Program starts at addr 10
                      ORG      3H
04 50                 JMP      50H           ; Interrupt jump vector
                      ORG      10H           ; Main program
99 FE                 ANL      P1, #0FEH     ; Chip select
81                    MOVX     A, @R1        ; Read in the 1st data
                                             ; to reset the intr
89 01      START:     ORL      P1, #1        ; Set port pin high
B8 20                 MOV      R0, #20H      ; Data address
B9 FF                 MOV      R1, #0FFH     ; Dummy address
BA 10                 MOV      R2, #10H      ; Counter for 16 bytes
23 FF      AGAIN:     MOV      A, #0FFH      ; Set ACC for intr loop
99 FE                 ANL      P1, #0FEH     ; Send CS (bit 0 of P1)
91                    MOVX     @R1, A        ; Send WR out
05                    EN       I             ; Enable interrupt
96 21      LOOP:      JNZ      LOOP          ; Wait for interrupt
EA 1B                 DJNZ     R2, AGAIN     ; If 16 bytes are read
00                    NOP                    ; go to user's program
00                    NOP
                      ORG      50H
81         INDATA:    MOVX     A, @R1        ; Input data, CS still low
A0                    MOV      @R0, A        ; Store in memory
18                    INC      R0            ; Increment storage counter
89 01                 ORL      P1, #1        ; Reset CS signal
27                    CLR      A             ; Clear ACC to get out of
93                    RETR                   ; the interrupt loop
```



TL/H/5671-22

FIGURE 12. INS8073 Interface

SAMPLE PROGRAM FOR *FIGURE 12* — INS8073 INTERFACE

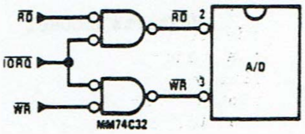
```
100   C = 16                        ; REM C is the 16 bytes counter
110   D = #13D0                     ; REM D points to data address
120   @ #3000 = A                   ; REM start A/D
130   A = STAT AND #20              ; REM wait until interrupt
140   IF A < > 0 THEN GO TO 130     ; REM from A/D
150   @ D = @ #3000                 ; REM input converted data
160   D = D + 1                     ; REM increment data address
170   C = C - 1                     ; REM check counter
180   IF C > 0 THEN GO TO 120       ; REM if 16 data have been read
190   RETURN                        ; REM return to main program
```

4.1.3 INS8073 Interface

The INS8073 allows users to program directly in Tiny Basic. DS1488/1489 driver/receiver chips are used for level buffering to communicate via RS-232. (For a detailed description of the INS8073 and the Tiny Basic, see INS8073 data sheet.) The ADC0801 is mapped into the memory space of the 8073 system (see *Figure 12*). A RAM of 1k bytes is provided in which the first 256 bytes are used by the Tiny Basic micro-interpreter. Address 3000 (Hex) is assigned to the A/D and the 16 converted data bytes are stored at external RAM locations from 13D0 to 13DF (Hex). STAT function is used to examine the interrupt signal from the A/D. A sample Tiny Basic subroutine is given in the sample program for *Figure 12* — INS8073 Interface.

4.2 Interfacing the Z-80

The Z-80 control bus is slightly different from that of the 8080. General $\overline{RD}$ and $\overline{WR}$ strobes are provided and separate memory request, $\overline{MREQ}$, and I/O request, $\overline{IORQ}$, signals are used which have to be combined with the generalized strobes to provide the equivalent 8080 signals. An advantage of operating the A/D in I/O space with the Z-80 is that the CPU will automatically insert one wait state (the $\overline{RD}$ and $\overline{WR}$ strobes are extended one clock period) to allow more time for the I/O devices to respond. Logic to map the A/D in I/O space is shown in *Figure 13*.



TL/H/5671-23

FIGURE 13. Mapping the A/D as an I/O Device for Use with the Z-80 CPU

Additional I/O advantages exist as software DMA routines are available and use can be made of the output data transfer which exists on the upper 8 address lines (A8 to A15) during I/O input instructions. For example, MUX channel selection for the A/D can be accomplished with this operating mode.

4.3 Interfacing 6800 Microprocessor Derivatives (6502, etc.)

The control bus for the 6800 microprocessor derivatives doe not use the $\overline{RD}$ and $\overline{WR}$ strobe signals. Instead it employs a single R/$\overline{W}$ line and additional timing, if needed, can be derived fom the $\phi 2$ clock. All I/O devices are memory mapped in the 6800 system, and a special signal, VMA, indicates that the current address is valid. *Figure 14* shows an interface schematic where the A/D is memory mapped in the 6800 system. For simplicity, the $\overline{CS}$ decoding is shown using ½ DM8092. Note that in many 6800 systems, an already decoded $\overline{4.5}$ line is brought out to the common bus at pin 21. This can be tied directly to the $\overline{CS}$ pin of the A/D, provided that no other devices are addressed at HX ADDR: 4XXX or 5XXX.

The following subroutine essentially performs the same function as in the case of the 8080A interface and it can be called from anywhere in the user's program.

In *Figure 15* the ADC0801 series is interfaced to the M6800 microprocessor through (the arbitrarily chosen) Port B of the MC6820 or MC6821 Peripheral Interface Adapter, (PIA). Here the $\overline{CS}$ pin of the A/D is grounded since the PIA is already memory mapped in the M6800 system and no $\overline{CS}$ decoding is necessary. Also notice that the A/D output data lines are connected to the microprocessor bus under program control through the PIA and therefore the A/D $\overline{RD}$ pin can be grounded.

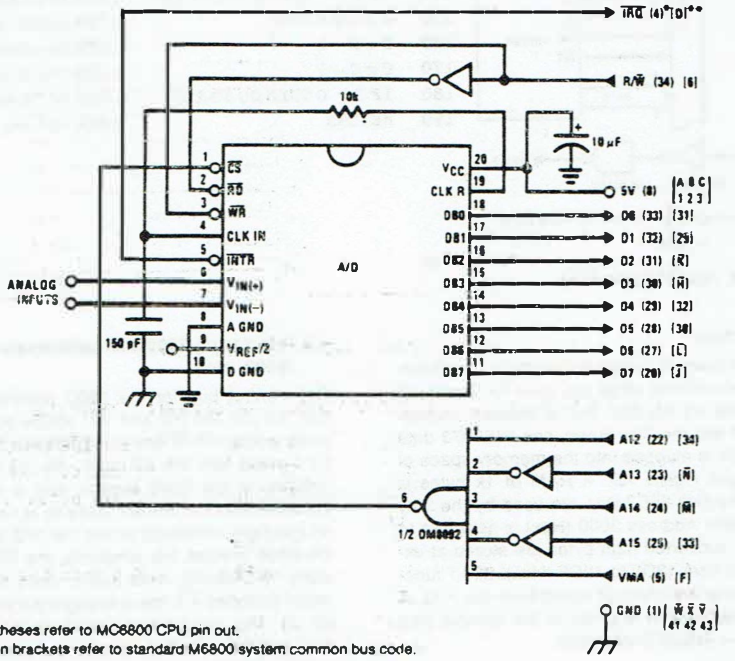
A sample interface program equivalent to the previous one, is shown below *Figure 15*. The PIA Data and Control Registers of Port B are located at HEX addresses 8006 and 8007, respectively.

5.0 GENERAL APPLICATIONS

The following applications show some interesting uses for the A/D. The fact that one particular microprocessor is used is not meant to be restrictive. Each of these application circuits would have its counterpart using any microprocessor which is desired.

5.1 Multiple ADC0801 Series to MC6800 CPU Interface

To transfer analog data from several channels to a single microprocessor system, a multiple converter scheme presents several advantages over the conventional multiplexer single-converter approach. With the ADC0801 series, the differential inputs allow individual span adjustment for each channel. Furthermore, all analog input channels are sensed simultaneously, which essentially divides the microprocessor's total system servicing time by the number of channels, since all conversions occur simultaneously. This scheme is shown in *Figure 16*.



Note 1: Numbers in parentheses refer to MC6800 CPU pin out.

Note 2: Number or letters in brackets refer to standard M6800 system common bus code.

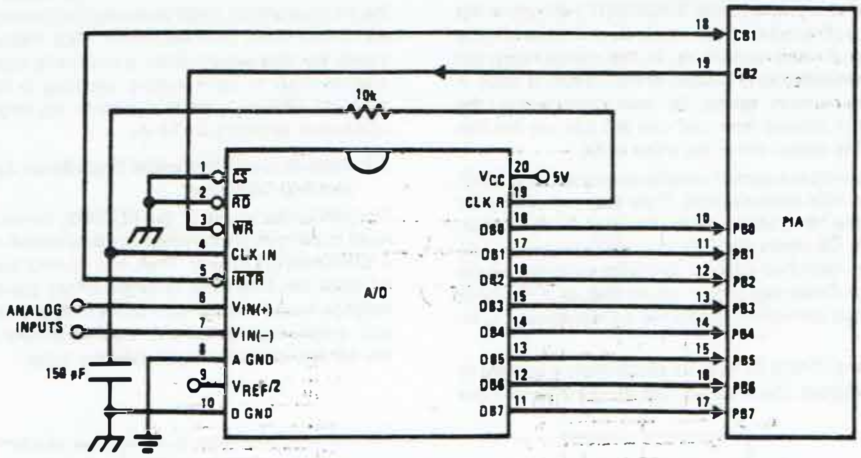
TL/H/5671-24

FIGURE 14. ADC0801-MC6800 CPU Interface

SAMPLE PROGRAM FOR *FIGURE 14* ADC0801-MC6800 CPU INTERFACE

```
0010   DF 36                 STX    TEMP2     ; Save contents of X
0012   CE 00 2C              LDX    #$002C    ; Upon IRQ low CPU
0015   FF FF F8              STX    $FFF8     ; jumps to 002C
0018   B7 50 00              STAA   $5000     ; Start ADC0801
001B   0E                    CLI
001C   3E          CONVRT    WAI              ; Wait for interrupt
001D   DE 34                 LDX    TEMP1
001F   8C 02 0F              CPX    #$020F    ; Is final data stored?
0022   27 14                 BEQ    ENDP
0024   B7 50 00              STAA   $5000     ; Restarts ADC0801
0027   08                    INX
0028   DF 34                 STX    TEMP1
002A   20 F0                 BRA    CONVRT
002C   DE 34       INTRPT    LDX    TEMP1
002E   B6 50 00              LDAA   $5000     ; Read data
0031   A7 00                 STAA   X         ; Store it at X
0033   3B                    RTI
0034   02 00       TEMP1     FDB    $0200     ; Starting address for
                                              ; data storage
0036   00 00       TEMP2     FDB    $0000
0038   CE 02 00    ENDP      LDX    #$0200    ; Reinitialize TEMP1
003B   DF 34                 STX    TEMP1
003D   DE 36                 LDX    TEMP2
003F   39                    RTS              ; Return from subroutine
                                              ; To user's program
```

Note 1: In order for the microprocessor to service subroutines and interrupts, the stack pointer must be dimensioned in the user's program.



TL/H/5671-25

FIGURE 15. ADC0801-MC6820 PIA Interface

SAMPLE PROGRAM FOR *FIGURE 15* ADC0801-MC6820 PIA INTERFACE

```
0010    CE 00 38    DATAIN    LDX     #$0038     ; Upon IRQ low CPU
0013    FF FF F8              STX     $FFF8      ; jumps to 0038
0016    B6 80 06              LDAA    PIAORB     ; Clear possible IRQ flags
0019    4F                    CLRA
001A    B7 80 07              STAA    PIACRB
001D    B7 80 06              STAA    PIAORB     ; Set Port B as input
0020    0E                    CLI
0021    C6 34                 LDAB    #$34
0023    86 3D                 LDAA    #$3D
0025    F7 80 07    CONVRT    STAB    PIACRB     ; Starts ADC0801
0028    B7 80 07              STAA    PIACRB
002B    3E                    WAI                ; Wait for interrupt
002C    DE 40                 LDX     TEMP1
002E    8C 02 0F              CPX     #$020F     ; Is final data stored?
0031    27 0F                 BEQ     ENDP
0033    08                    INX
0034    DF 40                 STX     TEMP1
0036    20 ED                 BRA     CONVRT
0038    DE 40       INTRPT    LDX     TEMP1
003A    B6 80 06              LDAA    PIAORB     ; Read data in
003D    A7 00                 STAA    X          ; Store it at X
003F    3B                    RTI
0040    02 00       TEMP1     FDB     $0200      ; Starting address for
                                                 ; data storage
0042    CE 02 00    ENDP      LDX     #$0200     ; Reinitialize TEMP1
0045    DF 40                 STX     TEMP1
0047    39                    RTS                ; Return from subroutine
                    PIAORB    EQU     $8006      ; To user's program
                    PIACRB    EQU     $8007
```

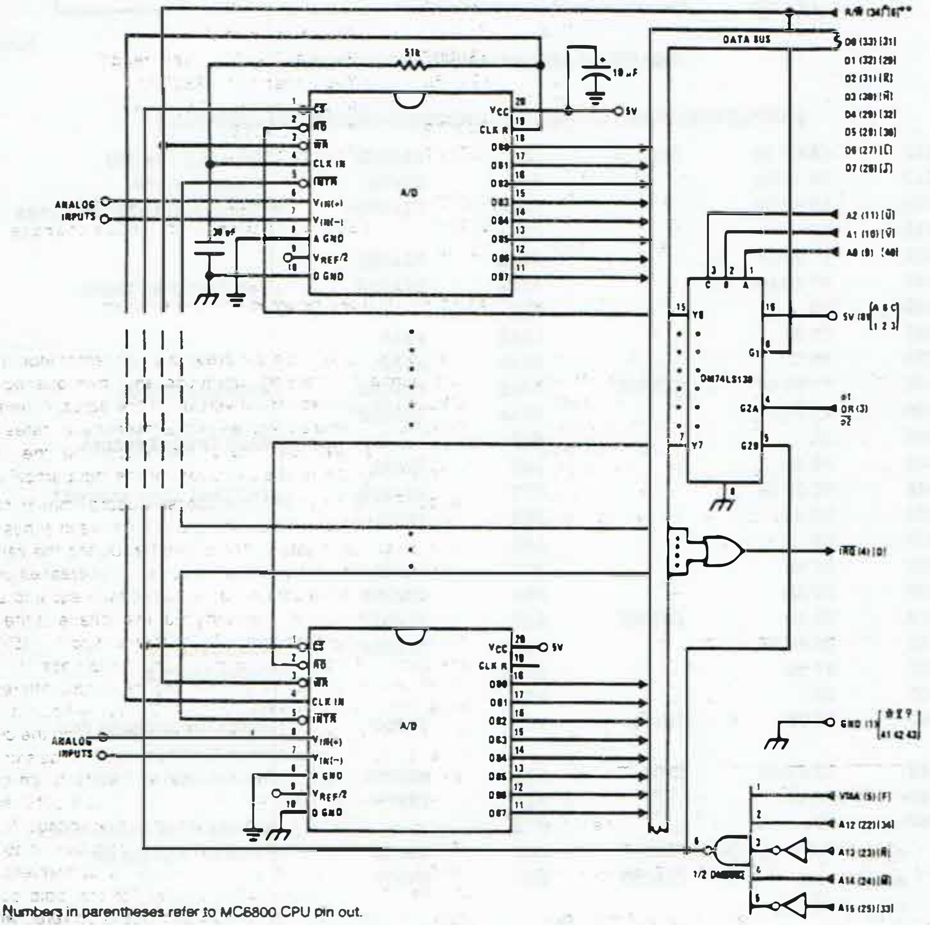
The following schematic and sample subroutine (DATA IN) may be used to interface (up to) 8 ADC0801's directly to the MC6800 CPU. This scheme can easily be extended to allow the interface of more converters. In this configuration the converters are (arbitrarily) located at HEX address 5000 in the MC6800 memory space. To save components, the clock signal is derived from just one RC pair on the first converter. This output drives the other A/Ds.

All the converters are started simultaneously with a STORE instruction at HEX address 5000. Note that any other HEX address of the form 5XXX will be decoded by the circuit, pulling all the $\overline{CS}$ inputs low. This can easily be avoided by using a more definitive address decoding scheme. All the interrupts are ORed together to insure that all A/Ds have completed their conversion before the microprocessor is interrupted.

The subroutine, DATA IN, may be called from anywhere in the user's program. Once called, this routine initializes the CPU, starts all the converters simultaneously and waits for the interrupt signal. Upon receiving the interrupt, it reads the converters (from HEX addresses 5000 through 5007) and stores the data successively at (arbitrarily chosen) HEX addresses 0200 to 0207, before returning to the user's program. All CPU registers then recover the original data they had before servicing DATA IN.

5.2 Auto-Zeroed Differential Transducer Amplifier and A/D Converter

The differential inputs of the ADC0801 series eliminate the need to perform a differential to single ended conversion for a differential transducer. Thus, one op amp can be eliminated since the differential to single ended conversion is provided by the differential input of the ADC0801 series. In general, a transducer preamp is required to take advantage of the full A/D converter input dynamic range.



Note 1: Numbers in parentheses refer to MC6800 CPU pin out.

Note 2: Numbers of letters in brackets refer to standard M6800 system common bus code.

TL/H/5671-26

FIGURE 16. Interfacing Multiple A/Ds in an MC6800 System

SAMPLE PROGRAM FOR *FIGURE 16* INTERFACING MULTIPLE A/Ds IN AN MC6800 SYSTEM

```
ADDRESS  HEX CODE            MNEMONICS               COMMENTS
0010     DF 44      DATAIN   STX     TEMP           ; Save Contents of X
0012     CE 00 2A            LDX     #$002A         ; Upon IRQ LOW CPU
0015     FF FF F8            STX     $FFF8          ; Jumps to 002A
0018     B7 50 00            STAA    $5000          ; Starts all A/D's
001B     0E                  CLI
001C     3E                  WAI                    ; Wait for interrupt
001D     CE 50 00            LDX     #$5000
0020     DF 40               STX     INDEX1         ; Reset both INDEX
0022     CE 02 00            LDX     #$0200         ; 1 and 2 to starting
0025     DF 42               STX     INDEX2         ; addresses
0027     DE 44               LDX     TEMP
0029     39                  RTS                    ; Return from subroutine
002A     DE 40      INTRPT   LDX     INDEX1         ; INDEX1 → X
002C     A6 00               LDAA    X              ; Read data in from A/D at X
002E     08                  INX                    ; Increment X by one
002F     DF 40               STX     INDEX1         ; X → INDEX1
0031     DE 42               LDX     INDEX2         ; INDEX2 → X
0033     A7 00               STAA    X              ; Store data at X
0035     8C 02 07            CPX     #$0207         ; Have all A/D's been read?
0038     27 05               BEQ     RETURN         ; Yes: branch to RETURN
003A     08                  INX                    ; No: increment X by one
003B     DF 42               STX     INDEX2         ; X → INDEX2
003D     20 EB               BRA     INTRPT         ; Branch to 002A
003F     3B         RETURN   RTI
0040     50 00      INDEX1   FDB     $5000          ; Starting address for A/D
0042     02 00      INDEX2   FDB     $0200          ; Starting address for data storage
0044     00 00      TEMP     FDB     $0000
```

Note 1: In order for the microprocessor to service subroutines and interrupts, the stack pointer must be dimensioned in the user's program.

For amplification of DC input signals, a major system error is the input offset voltage of the amplifiers used for the preamp. *Figure 17* is a gain of 100 differential preamp whose offset voltage errors will be cancelled by a zeroing subroutine which is performed by the INS8080A microprocessor system. The total allowable input offset voltage error for this preamp is only 50 µV for ¼ LSB error. This would obviously require very precise amplifiers. The expression for the differential output voltage of the preamp is:

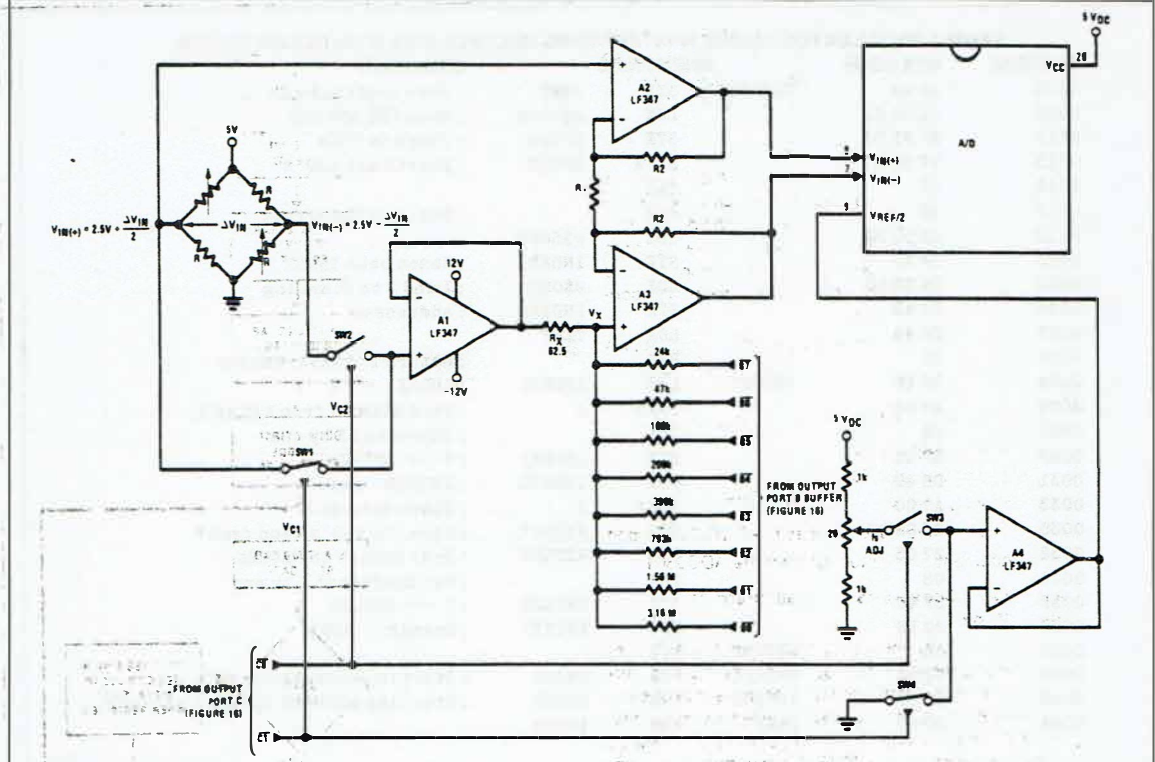
$$V_O = \underbrace{[V_{IN}(+) - V_{IN}(-)]}_{\text{SIGNAL}} \underbrace{\left[1 + \frac{2R2}{R1}\right]}_{\text{GAIN}} +$$

$$\underbrace{(V_{OS_2} - V_{OS_1} - V_{OS_3} \pm I_X R_X)}_{\text{DC ERROR TERM}} \underbrace{\left(1 + \frac{2R2}{R1}\right)}_{\text{GAIN}}$$

where I_X is the current through resistor R_X. All of the offset error terms can be cancelled by making $\pm I_X R_X = V_{OS1} + V_{OS3} - V_{OS2}$. This is the principle of this auto-zeroing scheme.

The INS8080A uses the 3 I/O ports of an INS8255 Programable Peripheral Interface (PPI) to control the auto zeroing and input data from the ADC0801 as shown in *Figure 18*. The PPI is programmed for basic I/O operation (mode 0) with Port A being an input port and Ports B and C being output ports. Two bits of Port C are used to alternately open or close the 2 switches at the input of the preamp. Switch SW1 is closed to force the preamp's differential input to be zero during the zeroing subroutine and then opened and SW2 is then closed for conversion of the actual differential input signal. Using 2 switches in this manner eliminates concern for the ON resistance of the switches as they must conduct only the input bias current of the input amplifiers.

Output Port B is used as a successive approximation register by the 8080 and the binary scaled resistors in series with each output bit create a D/A converter. During the zeroing subroutine, the voltage at V_X increases or decreases as required to make the differential output voltage equal to zero. This is accomplished by insuring that the voltage at the output of A1 is approximately 2.5V so that a logic "1" (5V) on any output of Port B will source current into node V_X thus raising the voltage at V_X and making the output differential more negative. Conversely, a logic "0" (0V) will pull current out of node V_X and decrease the voltage, causing the differential output to become more positive. For the resistor values shown, V_X can move ±12 mV with a resolution of 50 µV which will null the offset error term to ¼ LSB of full-scale for the ADC0801. It is important that the voltage levels which drive the auto-zero resistors be constant. Also, for symmetry, a logic swing of 0V to 5V is convenient. To achieve this, a CMOS buffer is used for the logic output signals of Port B and this CMOS package is powered with a stable 5V source. Buffer amplifier A1 is necessary so that it can source or sink the D/A output current.



Note 1: R2 = 49.5 R1

Note 2: Switches are CD4066BC CMOS analog switches.

Note 3: The 9 resistors used in the auto-zero section can be ±5% tolerance.

FIGURE 17. Gain of 100 Differential Transducer Preamp

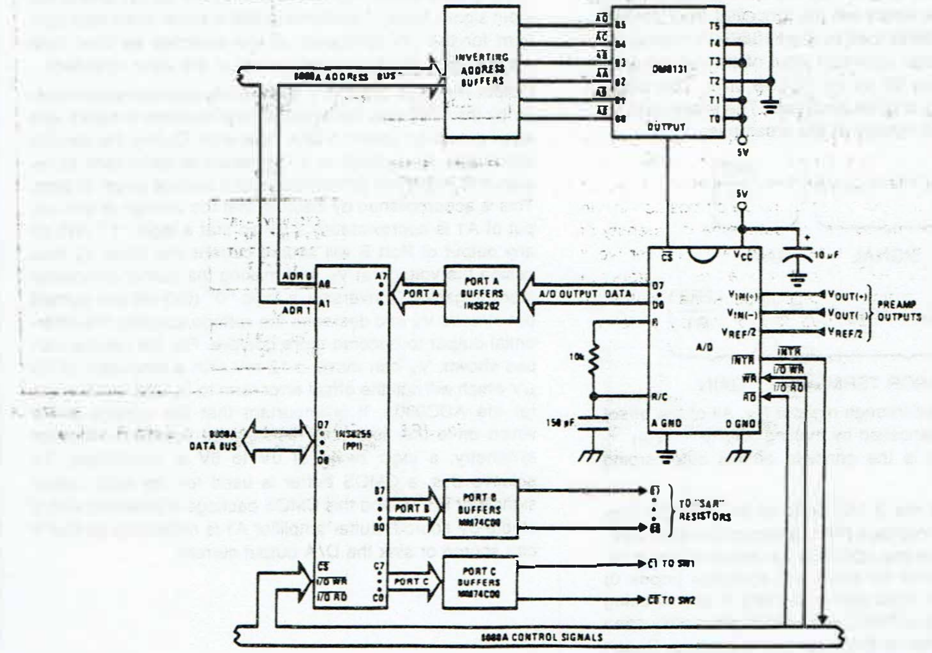


FIGURE 18. Microprocessor Interface Circuitry for Differential Preamp

TL/H/5671-27

A flow chart for the zeroing subroutine is shown in *Figure 19.* It must be noted that the ADC0801 series will output an all zero code when it converts a negative input [$V_{IN}(-) \geq V_{IN}(+)$]. Also, a logic inversion exists as all of the I/O ports are buffered with inverting gates.

Basically, if the data read is zero, the differential output voltage is negative, so a bit in Port B is cleared to pull V_X more negative which will make the output more positive for the next conversion. If the data read is not zero, the output voltage is positive so a bit in Port B is set to make V_X more positive and the output more negative. This continues for 8 approximations and the differential output eventually converges to within 5 mV of zero.

The actual program is given in *Figure 20.* All addresses used are compatible with the BLC 80/10 microcomputer system. In particular:

Port A and the ADC0801 are at port address E4

Port B is at port address E5

Port C is at port address E6

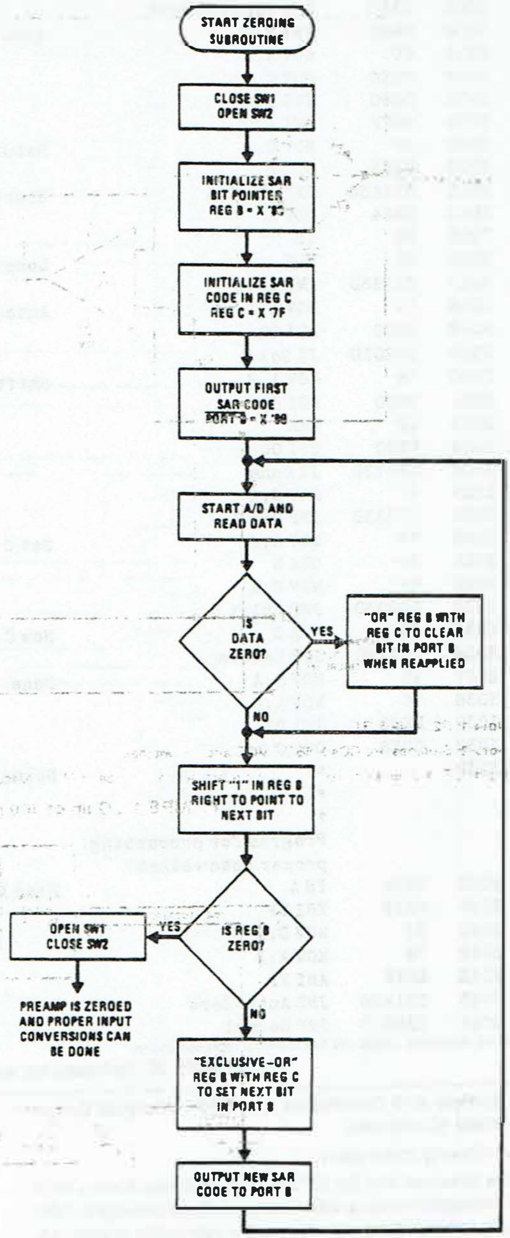
PPI control word port is at port address E7

Program Counter automatically goes to ADDR:3C3D upon acknowledgement of an interrupt from the ADC0801

5.3 Multiple A/D Converters In a Z-80 Interrupt Driven Mode

In data acquisition systems where more than one A/D converter (or other peripheral device) will be interrupting program execution of a microprocessor, there is obviously a need for the CPU to determine which device requires servicing. *Figure 21* and the accompanying software is a method of determining which of 7 ADC0801 converters has completed a conversion ($\overline{INTR}$ asserted) and is requesting an interrupt. This circuit allows starting the A/D converters in any sequence, but will input and store valid data from the converters with a priority sequence of A/D 1 being read first, A/D 2 second, etc., through A/D 7 which would have the lowest priority for data being read. Only the converters whose INT is asserted will be read.

The key to decoding circuitry is the DM74LS373, 8-bit D type flip-flop. When the Z-80 acknowledges the interrupt, the program is vectored to a data input Z-80 subroutine. This subroutine will read a peripheral status word from the DM74LS373 which contains the logic state of the $\overline{INTR}$ outputs of all the converters. Each converter which initiates an interrupt will place a logic "0" in a unique bit position in the status word and the subroutine will determine the identity of the converter and execute a data read. An identifier word (which indicates which A/D the data came from) is stored in the next sequential memory location above the location of the data so the program can keep track of the identity of the data entered.


START ZEROING SUBROUTINE
CLOSE SW1 OPEN SW2
INITIALIZE SAR BIT POINTER REG B = X '80
INITIALIZE SAR CODE IN REG C REG C = X '7F
OUTPUT FIRST SAR CODE PORT B = X '80
START A/D AND READ DATA
IS DATA ZERO?
YES
"OR" REG B WITH REG C TO CLEAR BIT IN PORT B WHEN REAPPLIED
NO
SHIFT "1" IN REG B RIGHT TO POINT TO NEXT BIT
IS REG B ZERO?
YES
OPEN SW1 CLOSE SW2
PREAMP IS ZEROED AND PROPER INPUT CONVERSIONS CAN BE DONE
NO
"EXCLUSIVE-OR" REG B WITH REG C TO SET NEXT BIT IN PORT B
OUTPUT NEW SAR CODE TO PORT B

TL/H/5671-28

FIGURE 19. Flow Chart for Auto-Zero Routine

```
3D00   3E90     MVI 90
3D02   D3E7     Out Control Port                               ; Program PPI
3D04   2601     MVI H 01              Auto-Zero Subroutine
3D06   7C       MOV A,H
3D07   D3E6     OUT C                                          ; Close SW1 open SW2
3D09   0680     MVI B 80                                       ; Initialize SAR bit pointer
3D0B   3E7F     MVI A 7F                                       ; Initialize SAR code
3D0D   4F       MOV C,A               Return
3D0E   D3E5     OUT B                                          ; Port B = SAR code
3D10   31AA3D   LXI SP 3DAA           Start                    ; Dimension stack pointer
3D13   D3E4     OUT A                                          ; Start A/D
3D15   FB       IE
3D16   00       NOP                   Loop                     ; Loop until INT asserted
3D17   C3163D   JMP Loop
3D1A   7A       MOV A,D               Auto-Zero
3D1B   C600     ADI 00
3D1D   CA2D3D   JZ Set C                                       ; Test A/D output data for zero
3D20   78       MOV A,B               Shift B
3D21   F600     ORI 00                                         ; Clear carry
3D23   1F       RAR                                            ; Shift "1" in B right one place
3D24   FE00     CPI 00                                         ; Is B zero? If yes last
3D26   CA373D   JZ Done                                        ; approximation has been made
3D29   47       MOV B,A
3D2A   C3333D   JMP New C
3D2D   79       MOV A,C               Set C
3D2E   B0       ORA B                                          ; Set bit in C that is in same
3D2F   4F       MOV C,A                                        ; position as "1" in B
3D30   C3203D   JMP Shift B
3D33   A9       XRA C                 New C                    ; Clear bit in C that is in
3D34   C30D3D   JMP Return                                     ; same position as "1" in B
3D37   47       MOV B,A               Done                     ; then output new SAR code.
3D38   7C       MOV A,H                                        ; Open SW1, close SW2 then
3D39   EE03     XRI 03                                         ; proceed with program. Preamp
3D3B   D3E6     OUT C                                          ; is now zeroed.
3D3D            •                     Normal
                •
                •
                Program for processing
                proper data values
3C3D   DBE4     IN A                  Read A/D Subroutine      ; Read A/D data
3C3F   EEFF     XRI FF                                         ; Invert data
3C41   57       MOV D,A
3C42   78       MOV A,B                                        ; Is B Reg = 0? If not stay
3C43   E6FF     ANI FF                                         ; in auto zero subroutine
3C45   C21A3D   JNZ Auto-Zero
3C48   C33D3D   JMP Normal
```

Note: All numerical values are hexadecimal representations.

FIGURE 20. Software for Auto-Zeroed Differential A/D

5.3 Multiple A/D Converters in a Z-80® Interrupt Driven Mode (Continued)

The following notes apply:

1) It is assumed that the CPU automatically performs a RST 7 instruction when a valid interrupt is acknowledged (CPU is in interrupt mode 1). Hence, the subroutine starting address of X0038.

2) The address bus from the Z-80 and the data bus to the Z-80 are assumed to be inverted by bus drivers.

3) A/D data and identifying words will be stored in sequential memory locations starting at the arbitrarily chosen address X 3E00.

4) The stack pointer must be dimensioned in the main program as the RST 7 instruction automatically pushes the PC onto the stack and the subroutine uses an additional 6 stack addresses.

5) The peripherals of concern are mapped into I/O space with the following port assignments:

HEX PORT ADDRESS	PERIPHERAL
00	MM74C374 8-bit flip-flop
01	A/D 1
02	A/D 2
03	A/D 3
04	A/D 4
05	A/D 5
06	A/D 6
07	A/D 7

This port address also serves as the A/D identifying word in the program.

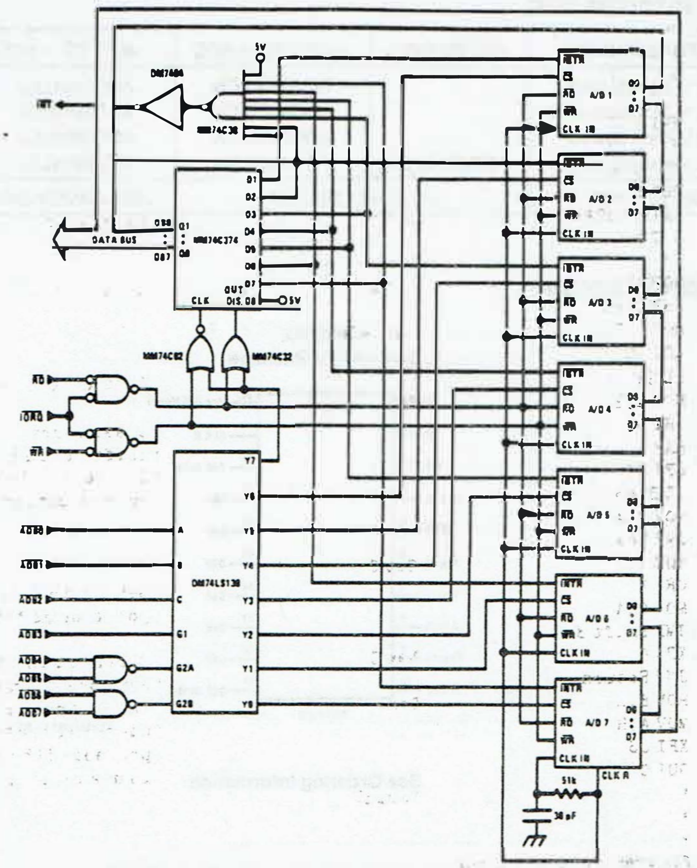


FIGURE 21. Multiple A/Ds with Z-80 Type Microprocessor

INTERRUPT SERVICING SUBROUTINE

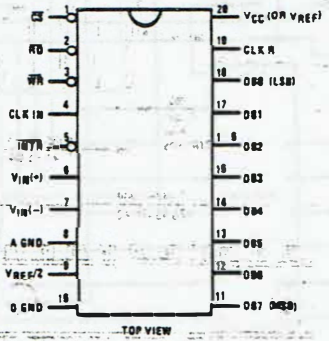
```
                           SOURCE
LOC   OBJ CODE            STATEMENT                            COMMENT
0038  E5                 PUSH HL          ; Save contents of all registers affected by
0039  C5                 PUSH BC          ; this subroutine.
003A  F5                 PUSH AF          ; Assumed INT mode 1 earlier set.
003B  21 00 3E           LD (HL),X3E00    ; Initialize memory pointer where data will be stored.
003E  0E 01              LD C, X01        ; C register will be port ADDR of A/D converters.
0040  D300               OUT X00, A       ; Load peripheral status word into 8-bit latch.
0042  DB00               IN A, X00        ; Load status word into accumulator.
0044  47                 LD B,A           ; Save the status word.
0045  79          TEST   LD A,C           ; Test to see if the status of all A/D's have
0046  FE 08              CP, X08          ; been checked. If so, exit subroutine
0048  CA 60 00           JPZ, DONE
004B  78                 LD A,B           ; Test a single bit in status word by looking for
004C  1F                 RRA              ; a "1" to be rotated into the CARRY (an INT
004D  47                 LD B,A           ; is loaded as a "1"). If CARRY is set then load
004E  DA 5500            JPC, LOAD        ; contents of A/D at port ADDR in C register.
0051  0C          NEXT   INC C            ; If CARRY is not set, increment C register to point
0052  C3 4500            JP,TEST          ; to next A/D, then test next bit in status word.
0055  ED 78       LOAD   IN A, (C)        ; Read data from interrupting A/D and invert
0057  EE FF              XOR FF           ; the data.
0059  77                 LD (HL),A        ; Store the data
005A  2C                 INC L
005B  71                 LD (HL),C        ; Store A/D identifier (A/D port ADDR).
005C  2C                 INC L
005D  C3 51 00           JP,NEXT          ; Test next bit in status word.
0060  F1          DONE   POP AF           ; Re-establish all registers as they were
0061  C1                 POP BC           ; before the interrupt.
0062  E1                 POP HL
0063  C9                 RET              ; Return to original program
```

Ordering Information

TEMPERATURE RANGE		0°C TO 70°C	−40°C TO +85°C	−40°C TO +85°C	−55°C TO +125°C
ERROR	± 1/4 Bit Adjusted		ADC0801LCN	ADC0801LCJ	ADC0801LJ
	± 1/2 Bit Unadjusted		ADC0802LCN	ADC0802LCJ	ADC0802LJ
	± 1/2 Bit Adjusted		ADC0803LCN	ADC0803LCJ	
	± 1 Bit Unadjusted	ADC0804LCN	ADC0805LCN	ADC0804LCJ	
PACKAGE OUTLINE		N20A-MOLDED DIP		J20A-CAVITY DIP	J20A-CAVITY DIP

Connection Diagram

ADC080X
Dual-In-Line Package

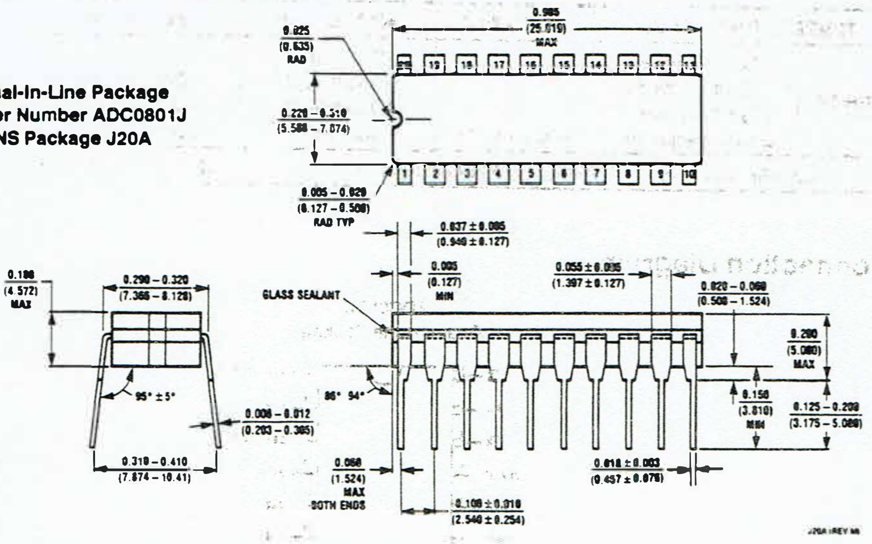


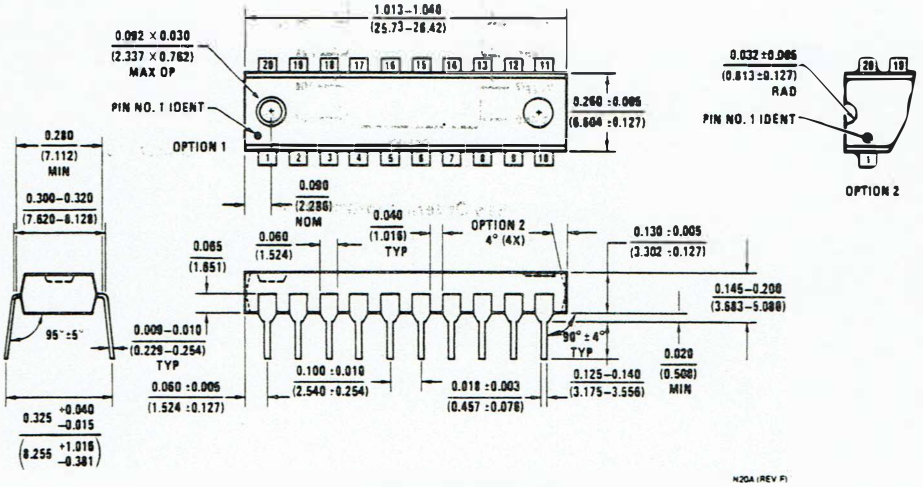
TL/H/5671–30

See Ordering Information

Physical Dimensions inches (millimeters)

Dual-In-Line Package
Order Number ADC0801J
NS Package J20A





Dual-In-Line Package
Order Number ADC0801N
NS Package N20A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: (408) 721-5000
TWX: (910) 339-9240

National Semiconductor GmbH
Westendstrasse 193-195
D-8000 Munchen 21
West Germany
Tel: (089) 5 70 95 01
Telex: 522772

NS Japan Ltd.
4-403 Ikebukuro,
Toshima-ku,
Tokyo 171, Japan
Tel: (03) 988-2131
FAX: 011-81-3-988-1700

National Semiconductor Hong Kong Ltd.
Southeast Asia Marketing
Austin Tower, 4th Floor
22-26 Austin Avenue
Tsimshatsui, Kowloon, H.K.
Tel: 3-7231290, 3-7243645
Cable: NSSEAMKTG
Telex: 52996 NSSEA HX

National Semicondutores Do Brasil Ltda.
Av. Brig. Faria Lima, 830
8 Andar
01452 Sao Paulo, SP, Brasil
Tel: (55/11) 212-5066
Telex: 391-1131931 NSBR BR

National Semiconductor (Australia) PTY, Ltd.
21/3 High Street
Bayswater, Victoria 3153
Tel: (03) 729-6333
Telex: AA32096

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.

INDEX

A

Active State 2-19
Actuators 5-68
ADDRESS
    Bus 3-54
    Code 3-7
    Decoder 3-55
    Decoding 3-53
Algorithm 4-72
Alternating Current B-11
Analog Signal 1-7
ANALOG-TO-DIGITAL
    Conversion 6-18
    Conversion Time 6-26
    Integration 6-20
    Linearity 6-26
    Resolution 6-25
    Accuracy 6-26
    Successive Approximation 6-20
    Transfer Function 6-24
    V/F Converters 6-22
AND Gate 1-17
AND Plane 3-83
AND/OR Design 2-57
ASCII 1-21, 24, A-11
Asynchronous 7-11

B

Back EMF 5-68
Backplane Board 1-40
Base Region B-20
Base Value A-5
Baud Rate 4-63
BCD A-8
BCD Counter 4-50, 58
BCD-to-Seven Segment Decoder/Driver 5-37
Bed of Nails 6-69
BINARY
    Addition A-19
    Coded Decimal A-8
    Display Mode 7-32
    Numbers A-5
    Point A-5
    Subtraction A-22
Binary-to-Hex A-18
Bipolar 1-43, 44
Bipolar Transistor B-20
Bit A-8
Black-Box Concept 2-9
Blanking 5-38
Boolean Expression 2-21
Boolean Laws 2-29
Bounce Regions 5-10
Buffer 1-56
Buffer Register 4-14, 32
Bugs Combinational Logic 7-6
Bugs Sequential Logic 7-11
Bus 2-58, 3-54
Bypass Capacitor 7-7

C

Capacitor B-15
Card Cage 1-40
Cascade 5-39
Channel Selector 3-6
Chassis 6-78
Circuit Repair 7-45
Clear 4-13, 18
CLOCK
    Circuit 4-9
    Circuit Testing 7-40
    Excessive Rates 7-15
    Skew 7-14
CMOS 1-53, 54
CMOS Decoder/Driver 5-43

CMOS to TTL 1-56
Collector Region B-20
COMBINATIONAL LOGIC
  Circuit 2-3
  Bugs 7-6
  Troubleshooting 7-40
Common Anode Display 2-22
Common Laws of Boolean Algebra 2-29
Complete Decoding 3-56
COMPUTER AIDED DESIGN
  Logic Design Using CAD 6-71
  Logic Simulation 6-72
  PC Board Design 6-73
  Software Breadboard 6-72
  Software Prototyping 6-70
Control Bus 3-54
Control Devices 5-61
Control Line 1-24
Controlled Buffer Register 4-33
Controlled Inverter 1-31
Counters 4-45
Current B-3
Cycle B-12

D

D Flip-Flop 4-9
D Flip-Flop Design 4-76
Darlington 5-77
Data Bus 3-54
Data Display Mode 7-32
Data Distributor 3-56
Data Selector 3-6
Debouncing Switches 5-10
Debugging 3-74
Decade Counter 4-50, 58
Decimal Numbers A-4
Decimal-to-Hex A-15
DECODER/DECODING
  And Driving Segmented Displays 5-37
  Complete 3-56
  IC devices 3-44
  Partial 3-56
  Theory 2-12, 3-41, 5-15
  Types 3-41
DeMorgan's First Theorem 1-18
DeMorgan's Second Theorem 1-30
Demultiplexer 3-56
Digital Logic Optoisolator 5-78
Digital Signal 1-9
DIGITAL-TO-ANALOG
  Accuracy 6-16
  Conversion 6-6
  Input Codes 6-10
  Linearity 6-16
  Resolution 6-14
  Settling Time 6-16
  Step Size 6-9
Diode B-18
Diode Rectifier 6-32
Diode ROM 3-65
Direct Current B-3
Discrete Transistor 1-37
DISPLAYS
  Blanking 5-38
  Devices 5-24
  Gas Discharge 5-32
  Incandescent 5-31
  LED 3-31
  Liquid Crystal 5-34, 45
  Multiplexing 5-49
  Segmented 5-30, 37
  Vacuum Fluorescent 5-33
Divide-by-12 Counter 4-59
Divide-by-16 Counter 4-60
Documentation 7-37
Don't Cares 2-47, 4-80

Dot Matrix Displays 5-52
Drilling 6-60
Drive Transistor 5-42
Dry Transfer Masking 6-63
Dual In-Line 1-38
Dubble-Dabble A-6
Duty Cycle 4-17, B-14
Dynamic Testing 7-45

E

EAROM 3-74
ECL 1-44, 50
Edge Connector 1-40
Edge Triggering 4-12
EEROM 3-74
Electrical Control Devices 5-68
Electrical Noise 1-35
Electrically Alterable ROM 3-74
Electrically Erasable ROM 3-74
Electro-Mechanical Relays 5-69
Electrodeless Tint 6-67
Electronic Term Prefixes B-5
Emitter-Coupled Logic 1-44, 50
Emitter Region B-20
Encoding 5-18
EPROM 3-72
Erasable/Programmable ROM 3-72
Etching 6-66
Even Parity 1-21
Excessive Clock Rates 7-15

F

Fall Time 1-32
Fanout 1-36
FET Optoisolator 5-78
Flip-Flop 4-6
Flat Pack 1-38
Floating Gate 3-72
Floating Inputs 7-7
Forward Bias B-19
Frequency B-12
Fully Encoded 5-19
Fundamental Product 2-21
Fuse Link 3-71

G

Gas Discharge Displays 5-32
GATE
    AND 1-17
    Conversion Chart 2-60
    NAND 1-19
    NOR 1-15
    OR 1-13
    XNOR 1-23
    XOR 1-20
Glitch 4-74, 7-6
Gray Code 4-74
Ground Loop 6-79
Grounding 6-78
Grouping 2-34

H

Hang-Up States 7-16
Hardware Prototyping 6-46
Hertz B-13
Hexadecimal A-13
Hex Counter 4-60
Hex-to-Binary A-17
High Level Noise Margin 1-46
High Speed TTL (74H Series) 1-49
HMOS 1-53
Hold Time 4-19

I

I/O Definition 2-18
IC DEVICES
  Description 1-31
  14511 5-43
  14543 5-47
  555 4-23
  74112 4-22
  74113 4-22
  74114 4-22
  7473 4-21
  7474 4-20
  7475 4-20
  7476 4-21
  7490 4-58
  7492 4-59
  7493 4-60
  74LS194 4-44
  74LS374 4-37
  74LS377 4-37
  74LS395 4-44
  74LS47 5-37
  74LS48 5-41
  75461 5-62
  CD4511 5-43
  CD4543 5-47
  Timer 4-23
IIL 1-44, 52
Incandescent Displays 5-31
Incandescent Lamps 5-24
Input Detect 3-41
Inspection and Quality Control 6-68
Integrated Circuit 1-31
Integrated-Injection Logic 1-44, 52
Integration 6-20
Inverter 1-11
IRED 5-74

J

JK Flip-Flop 4-15
JK Flip-Flop Design 4-79

K

Karnaugh Mapping 2-33
Keyboard 5-15
Keyboard Encoder IC 5-18
Kludging 6-52

L

Large-Scale Integration 1-37
Latch 4-10, 14, 32
Laws of Boolean Algebra 2-29
Leading Edge 1-32
LED Display 2-12
LEDs 5-28, 31
Level Triggered 4-11
Linear Power Supply 6-32
Liquid Crystal Displays 5-34
LOGIC ANALYZER
  Binary Display 7-32
  Data Display 7-32
  Map Mode 7-32
  Signature 7-32
  Timing Mode 7-32
LOGIC
  Clips 7-25
  Definition 2-19
  Gate Conversion Chart 2-60
  Monitors 7-26
  Probes 7-22
  Pulsers 7-26
Low Level Noise Margin 1-46
Low Power Schottky TTL (74LS Series) 1-50
Low Power TTL (74L Series) 1-48
LSI 1-37

M

Map Mode 7-32
Mask-Programmed ROM 3-69
Mechanical Switches 5-7
MECL 1-52
Medium-Scale Integration 1-37
Metallization Mask 3-70
Mod 4-48
Modem 4-63
Modulus 4-48
Monolithic 1-43
MOS 1-43, 53
MOS Precautions 1-59
MOSFET 1-53, 3-72
Mother Board 1-40
Motorola Emitter-Coupled Logic 1-52
MSI 1-37
Multimeters 7-29
Multiplexed Displays 5-49
Multiplexer IC Devices 3-11
Multiplexers 3-6

N

n – 1 Multiplexer Solution 3-18
n – 2 Multiplexer Solution 3-24
NAND Gate 1-19
NAND/NOR/XOR Design 2-60
Natural Modulus 4-48
Negative Logic 2-19
Nibble A-8
NMOS 1-53
Noise Immunity 1-35
Noise Margin 1-46
NOR Gate 1-15
Numerical Form 2-24

O

Obvious Solution 3-16
Odd Parity 1-21
Odd Parity System 1-24
Ohm's Law B-9
Ohms B-6
One's Complement A-22
Open Collector Device 1-58
Open Collector Drivers 5-26
Optical Comparator 6-68
Optical Isolator 6-80
Optocouplers 5-74
OPTOISOLATOR
- Digital Logic 5-78
- FET 5-78
- Photo-Darlington 5-77
- Theory 5-74

OR Gate 1-13
OR Plane 3-83
Oscilloscopes 7-30

P

Packing Density 1-36
Parallel Input Shift Registers 4-41
Parallel-to-Serial Conversion 4-43
Parity 1-21, 24
Partial Decoding 3-56
PCs 1-39
Pen Applicator 6-62
Perfboard 6-52
Period B-12
Periodic Signals B-12
Peripheral Power Drivers 5-61
Phase Comparator 1-27
Photo Masking 6-63
Photo-Darlington 5-77

Photolithography 1-37
Photomask 6-63
Photoresist 6-64
PLA 3-83
PLD 3-83
Plotting 2-34
PMOS 1-53
PN Junction B-18, B-20
Positional Weight A-4
Positive Logic 2-19
Potentiometer B-8
Power B-10
Power Dissipation 1-33
POWER SUPPLY
  Glitches 7-6
  Linear 6-32
  Module 6-40
  Switching 6-34
  Testing 7-39
Power Transformer 6-32
Prefixes For Electronic Terms B-5
Preset 4-13, 18
Presettable Counters 4-52
Printed Circuit Board 1-39, 6-60
Product Development 2-8
Programmable Log Devices 3-83
Programmable Logic Arrays 3-83
Programmable ROM 3-71
PROM 3-71
PROM Burner 3-71
Propagation Delay Time 4-19, 7-9
Propagation Time 1-32
Prototype Masking 6-62
Pull-down Resistor 1-51
Pull-up Resistor 1-57

R

Race Condition 4-7
Rack 1-40
Radio Frequency Interference 6-77
RC Time Constant 5-13
Redesign and Alter 7-45
Register 3-65, 4-32
Register Address 3-66
Regulator Circuit 6-32
RELAYS
  Electro-Mechanical 5-69
  Solid State 5-66
Repairing the Circuit 7-45
Resist Pen 6-62
Resist Tape 6-62
Resistance B-5
Resistor Color Code B-7
Reverse Bias B-19
Ring Counter 4-56
Ripple Blanking 5-38
Ripple Counter 4-46
Rise Time 1-32
ROM 3-65
Rotate Left 4-57
Rotate Right 4-57
RS Flip-Flop 4-6
RS232C-to-TTL 5-80

S

Schmitt-Trigger 5-13
Schottky Diode 1-49
Schottky TTL (74S Series) 1-49
Segmented Displays 5-30
Sequential Circuit Design Algorithm 4-71
Sequential Circuit Troubleshooting 7-43
Sequential Logic 2-3
Sequential Logic Bugs 7-11
Serial Input Shift Registers 4-38
Serial-to-Parallel Conversion 4-40
Setup Time 4-19
Shielding 6-77
Shift Register 4-32, 38

Signature 7-32
Silk Screen Masking 6-66
Simplification Using Boolean Algebra 2-29
Sink Current 2-19
Sinking Current 5-25
SLSI 1-37
Small-Scale Integration 1-37
Smoothing Filter 6-32
Snubber Circuit 5-68
Software Prototyping 6-70
Solder Wick 7-46
Soldering 6-67
Solderless Breadboard 6-47
Solenoids 5-73
Solid State Relays 5-66
SOP 2-22
Source Current 2-19
Speed-Power Product 1-34
SSI 1-37
SSI Design Algorithm 2-13
SSI Implementation 2-57
Standard TTL (7400 Series) 1-47
State Table 4-49, 75
Static State 1-54
Static Testing 7-45
Step Size 6-9
Straight Binary A-8
Subrack 1-40, 6-78
Successive Approximation 6-20
Sum-of-Product 2-22
Super Large-Scale Integration 1-37
Switch Debouncing 5-10
Switches 5-7
Switching Power Supply 6-34
Synchronous Counter 4-53

T

T^2 1-44
t_f 1-32
t_h 4-19
t_p 1-32
t_p 4-19
t_{PHL} 1-33
t_{PLH} 1-33
t_r 1-32
t_s 4-19
Tape Masking 6-62
Testing the Clock Circuit 7-40
Testing the Power Supply 7-39
Three State Buffering 3-68
Three-State Outputs 4-35
Time Constant B-16
Timer IC 4-23
Timing 7-11
Timing Diagram 1-24
Timing Mode 7-32
TO5 1-38
Toggle 4-16
Trailing Edge 1-32
Transfer Function 6-24
Transistor B-20
Transistor Arrays 5-65
Transistor Drive 5-42
Transistor-Transistor Logic 1-44
Tri-State Buffering 3-68
Tri-State Outputs 4-35
TRIGGERING
    Edge 4-12
    Level 4-11
Troubleshooting and Repair 7-37
Troubleshooting Combinational Logic 7-40
Troubleshooting Sequential Circuits 7-40
Truth Table 2-20
TTL (7400 Series) 1-47
TTL (74H Series) 1-49
TTL (74L Series) 1-48
TTL (74LS Series) 1-50
TTL 1-44
TTL-to-CMOS 1-57
TTL-to-RS232 5-80
Two's Complement A-22

U

Unclocked Inputs 7-11
Unencoded 5-19

V

V_{CC} 1-47
V_{DD} 1-54
V_{IH} 1-45
V_{IL} 1-45
V_{NH} 1-46
V_{NL} 1-46
V_{OH} 1-45
V_{OL} 1-45
V/F Converters 6-22
Vacuum Fluorescent Displays 5-33
VARIABLES
    Two-Input 2-33
    Three-Input 2-38
    Four-Input 2-40
Very Large-Scale Integration 1-37
VLSI 1-37
Voltage B-5

W

Wave Soldering 6-67
Wire List 6-56
Wire Wrapping 6-54
Wiring Delay 7-15
Word A-8
Word Selector 3-11

X

XNOR Gate 1-23
XOR Gate 1-20

Z

Zero-Voltage Switching 5-67